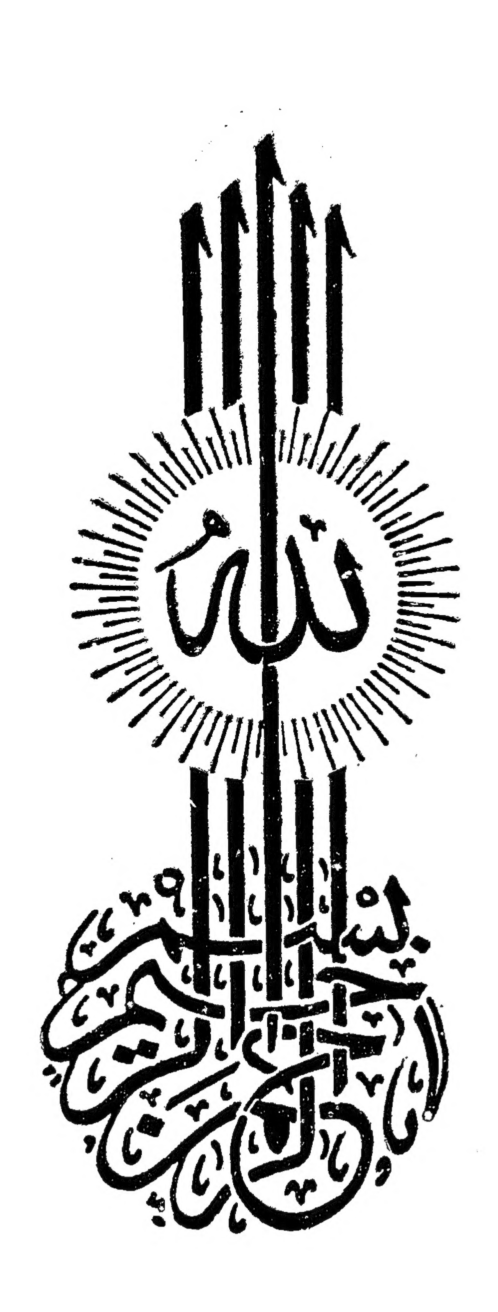

الله
بسم الله الرحمن الرحيم

# تطوير برنامج لتحليل الكلام باستخدام ماتلاب (MATLAB)

إعداد

**ظافر بن سعيد بن مناع العمري**

**رسالة مقدمة لاستكمال متطلبات درجة الماجستير**

**في الهندسة الكهربائية وهندسة الحاسبات**

**(هندسة إلكترونيات واتصالات كهربائية)**

**كلية الهندسة**
**جامعة الملك عبد العزيز**
**جدة – المملكة العربية السعودية**
**ذو القعدة ١٤٢٢ هـ - يناير ٢٠٠٢ م**

# إهداء

إلى والدي ووالدتي وزوجتي واخوتي الأعزاء

لكم مني كل الحب والتقدير والعرفان

# شكر و تقدير

الحمد لله والصلاة والسلام على أشرف الأنبياء والمرسلين ، سيدنا محمد وعلى آله وصحبه أجمعين قال تعالى : ﴿ **... لَئِن شَكَرتُمْ لأَزِيدَنَّكُمْ ...** ﴾ إبراهيم آية (٧) .

أتقدم بخالص الشكر والامتنان لوالدي و والدتي وزوجتي.

وأخص بالشكر أستاذي ومعلمي الفاضل سعادة الدكتور أحمد بن سعيد بالعمش على وقته الثمين وإشرافه المثمر.

كما أتقدم بالشكر لأعضاء هيئة التدريس بقسم الهندسة الكهربائية وهندسة الحاسبات. وأخص بالشكر أستاذي القدير سعادة الدكتور رباح بن واصل الظاهري.

# تطوير برنامج لتحليل الكلام باستخدام ماتلاب (MATLAB)

**ظافر بن سعيد بن مناع العمري**

## المستخلص

في هذه الرسالة تم عمل برنامج متكامل لتحليل الكلام، هذا البرنامج عمل تحت الماتلاب، ويشمل البرنامج العديد من التحليلات منها: منحنى الطاقة، الصورة الطيفية، واستخلاص معدل اهتزاز الحبال الصوتية والترددات الأساسية في الصوت وعملية فصل الصوت إلى مجهور ومهموس والتحليل الطيفي. كلاً من هذه العمليات التحليلية له الخيارات الخاصة به والعوامل المؤثرة فيه والتي تؤثر بدورها في النتائج.

هذا البرنامج أداة تدعم الباحثين في مجال علم اللغة وعلم تمثيل الأصوات. كما أن هذا البرنامج يطبق من قبل الأطباء الذين يعملون في عيادات أمراض الكلام، حيث يتم تحليل صوت المريض ومن ثم تدريبه للوصول إلى النتائج الصحيحة عن طريق عرض النتائج للمريض. وفي الآونة الأخيرة اقترح بعض علماء القرآن الكريم استعمال مثل هذا البرنامج كوسيلة لتصحيح القراءة والتجويد.

في هذه الرسالة تعرضت إلى كيفية توليد الكلام وإخراج الصوت عند الإنسان، كما تعرضت إلى عدد من الخوارزميات لمعالجة وتحليل الكلام والخلفيات الرياضية لها.

كما شملت الرسالة كيفية تصميم شاشة المستخدم وقوانين تصميمها، وكذلك دليل المستخدم لتوضيح كيفية استخدام البرنامج.

# تطوير برنامج لتحليل الكلام باستخدام ماتلاب (MATLAB)

ظافر بن سعيد بن مناع العمري

## الملخص

بحث الكثير في تحليل ومعالجة الكلام ومنهم ما قام به ستيفن كاندليس في عام ١٩٧٤م حيث قدَم خوارزمية لاستخلاص الثلاثة الترددات الأساسية الأولى للصوت. وفي ١٩٧٨م أثبت رابينار و شيفار كيف يمكن تطبيق تقنية معالجة الإشارات الرقمية على المشاكل المتعلقة باتصالات الكلام. في ١٩٩١م وصف دايفيد كروبساك طريقة لاستخلاص التردد الأساسي للصوت ولتصنيف الصوت إلى مجهور وغير مجهور. وفي عام ١٩٩٥م ألف جون ديللر و جون هانسين كتاب في معالجة الكلام عن طريق المعالجة الرقمية. وهناك العديد من الأبحاث نظراً لأهمية هذا المجال ووسع تطبيقاته.

في هذه الرسالة تم عمل برنامج متكامل لتحليل الكلام، هذا البرنامج عمل تحت الماتلاب، هذا البرنامج يمكن المستخدم من تسجيل مقطع من كلامه ومن ثم القيام بعمليات تحليل متعددة، لإستخلاص بعض خصائص الصوت. من هذه الخصائص تردد البتش ( Pitch frequency ) هو معدل اهتزاز الحبال الصوتية في الجهاز الصوتي عند الإنسان في الثانية الواحدة. خاصية أخرى الترددات الأساسية (Formant) وهي مضاعفات لتردد البتش وذلك نتيجة لمرور الهواء الخارج من الرئتين والحلق مسبباً اهتزاز الحبال الصوتية عبر الفم والأنف واللذان يمثلان مضخمان للتردد الرئيسي تردد البتش. من العمليات

التي يقوم بها البرنامج أيضاً فصل الصوت المجهور (Voiced) عن المهموس (Unvoiced) كما يقوم بعرض الصورة الطيفية (Spectrogram) لمقطع الكلام المختار. كل عملية تحليل في هذا البرنامج لها عدد من الخيارات أو العوامل (Options) والتي تعطي الفرصة لمستخدم البرنامج لتغيير أي من هذه العوامل والذي يعطي نتائج مختلفة تبين أثر هذا العامل على التحليل.

تعرضت في هذه الرسالة إلى عملية توليد وإخراج الصوت عند الإنسان ، حيث نرى قدرة الخالق سبحانه في إحكامه لهذا الإنسان وجميع أجهزته العضوية. الكلام الإنساني يتطلب تعاون ثلاثة أجهزة من أجهزة الإنسان الجسدية من أجل إخراج الأصوات، الجهاز التنفسي والجهاز الصوتي والجهاز النطقي. من أهم مكونات الجهاز الصوتي الوتران الصوتيان أو الحبلان الصوتيان ( Vocal cords ) وهما عبارة عن عضلة هرمية ونسيج مرن. كما أن الجهاز النطقي مكون من قسمين: قسم متحرك، ويضم: الرئتين،الوترين الصوتيين، الحنجرة،اللهاة، والطبق، والفك الأسفل، واللسان، والشفتين. القسم الثاني هو القسم الثابت ويضم: الجدار الخلفي للحلق، الغار، اللثة، والأسنان. ويصدر الكلام عندما تتقلص عضلات صدر الإنسان، وعضلات بطنه، فيضغط قفص الصدر، بفضل الحجاب الحاجز والعضلات الواقعة بين ريش الصدر، على الرئتين اللتين تدفعان مقداراً من الهواء -هواء الزفير- إلى الأعلى، حيث يجتاز الحنجرة- وهي مجموعة غضاريف يلتصق بأعلاها الوتران الصوتيان اللذان يتخذان وضعاً يختلف اختلافاً كلياً، في حالة التلفظ عما هو عليه في حالة التنفس- ويتحول هواء الزفير هذا، بعمل الوترين الصوتيين، إلى ذبذبات دورية وهي نواة الكلام. ثم يندفع التدفق الهوائي الصوتي من الحنجرة إلى الأعضاء التي تقع فوق الحنجرة، والتي تتألف من

الحلق والتجويف الأنفي، وتجويف الفم الذي يحتوي على الحنك وشراعه، وعلى اللسان، والأسنان، وينتهي بالشفتين... وتتلاعب هذه الأعضاء بالمادة الصوتية، وتكيفها، وتحولها إلى أصوات لغوية، أي تبرز العناصر الأساسية في الكلام، وتزيده وضوحاً ودقة، وتجرده من الشوائب التي قد تكون امتزجت به. فموضع النطق إذا، هو ما يسميه علماؤنا القدامى "مخرج الحرف". ويستطيع الإنسان إخراج مالا يحصى من الأصوات..لكن متكلمي كل لغة يكتفون عادةً بعدد محدود منها، يؤلفون منها أصوات لغتهم، أي كلامهم ، لان "اللغة أصوات يعبر بها كل قوم عن أغراضهم". ونحن كمهندسين اعتبرنا بعض هذه الأعضاء كمرشحات صوتية لها عوامل تمثلها وذلك مثل المجرى الصوتي الذي يبدأ بعد فتحة المزمار إلى الشفاه، التجويف الأنفي، وبعضها تم تمثيله كمولد وهي الرئتين والحنجرة (الحبال الصوتية) في حالة توليد الصوت المجهور ويمثل الصوت المهموس بتشويش.

صنفت الأصوات إلى صامتة (Consonants) وصائته (Vowels)، وذلك نتيجة دراسة طبيعة الأصوات، وصفاتها، ونتيجة وضع الأوتار الصوتية وكيفية مرور الهواء من الحلق إلى الفم والأنف .الأصوات الصائته تصدر دون أن يصطدم هواء الزفير بأي عائق وهي جوفيه لأنها تخرج من الجوف ، والصوائت كلها مجهورة. أما الأصوات الصامتة فهي الأصوات الناتجة أثناء النطق عن اصطدام الهواء بعائق وتكون مجهورة أو مهموسة وكل صوت مهموس هو صامت. الجهر هو انحباس مجرى النفس عند النطق بالصوت لقوته ، أي أن ضغط الهواء يكون مغلقاً، فيحدث ضغط هواء الزفير تذبذباً في الوترين الصوتيين فيصدر الصوت المجهور. والجهر يكون بانقباض فتحة المزمار (Glottis) و باقتراب الوترين الصوتيين أحدهما من الآخر. أما

الهمس، ضد الجهر وهو انطلاق هواء التنفس عند النطق بالصوت لضعفه، إذ ينطلق الهواء ولا يعوقه مروره في الحنجرة أي عائق فلا يتذبذب الوتران الصوتيان حيث يرتخيان ولا يهتزان.

كان الباب الثالث في هذه الرسالة الخلفية النظرية والرياضية لعمليات التحليل المستخدم في برنامج تحليل الكلام . تحليل ومعالجة الكلام يمكن أن يكون في المجال الزمني ( Time domain ) أو المجال الترددي ( Frequency domain ) ، حيث في البعد الزمني يتم التعامل مع الإشارة مباشرةً لاستخلاص بعض الحقائق والمميزات. أما في المجال الترددي فيتم التعامل مع تمثيل فورير للإشارة الصوتية . المجال الترددي يخدم في إظهار وتوضيح بعض خصائص الصوت والتي قد تكون غامضة أو أقل وضوحاً في الإشارة الأساسية قبل التحويل.

الفرض الأساسي في كل عمليات التحليل هو أن خصائص الإشارة الصوتية تتغير نسبياً ببطء مع الزمن. هذا الفرض قاد إلى عدد من طرق تحليل الزمن القصير (Short Time Analysis) ، والتي يتم فيها أخذ مقطع صغير من الصوت ثم تبدأ التحليلات عليه. من خلال حساب منحى الطاقة وجدنا الفرق الكبير بين المقاطع المجهورة والمقاطع المهموسة حيث يكون أعلى بكثير في المجهورة.

يطلق مصطلح "الحزم الصوتية" على الترددات المقواة أو المضخمة أو على مجموعة الترددات التي تشكل نوع الصوت أو طابعه، وتميزه من الأصوات الأخرى ذات الأنواع المختلفة وكل صوت من الصوائت يملك نغمة أساسية واثنتين على الأقل من الحزم الصوتية. والحزم الصوتية هي الترددات التي تضخم في المجرى الصوتي وتكون واضحة جداً في المجال الترددي نظر

الارتفاع قيمة الموجه عند هذه الترددات. وتعتمد الحزم الصوتية على شكل المجرى الصوتي والذي يتحدد بالمحسنات الصوتية كالسان والشفتين. الحصول على الحزم الصوتية يتم عن طريق استخلاص أول ثلاث قمم في الطيف لمقطع من الكلام يمثل حرف عله. في عملية استخلاص ترد البتش يجب أن يكون المقطع الكلامي مقطع مجهور لأن البتش تكون واضحة جداً في هذا المقطع. وترد البتش يقع بين ١٠٠ إلى ٣٠٠ هيرتز حيث تكون عادةً عالية عند الأطفال و النساء.

الباب الرابع يتحدث عن الأساسيات لتصميم شاشة المستخدم. التصميم يعني كل ما يعمل قبل البدء في كتابة البرنامج، من أهم مبادئ التصميم: البساطة، الاتساق والألفة أو الاعتيادية. كل كائن يظهر في شاشة المستخدم له معرف وحيد، والذي يساعد للرجوع إلى الكائن وتغيير خصائصه في أي وقت.

برنامج تحليل الكلام يتكون من الشاشة الرئيسية والتي تحتوي على تسع قوائم (menus) و محورين (axis). القوائم هي قائمة الملف، قائمة الأدوات ( Tool menu )، وقائمة الصورة الطيفية، وقائمة منحى الطاقة، وقائمة فصل الصوت المجهور عن المهموس، وقائمة تردد البتش، وقائمة منحى الطاقة، و قائمة التمثيل الترددي وأخيراً قائمة سماع الصوت ( Play menu ) . بعض هذه القوائم لها قوائم فرعية ( submenu ) . الداخل إلى هذا البرنامج هو الإشارة الصوتية عن طريق كرت الصوت والمكبر في الحسب الآلي. بعد عملية التسجيل تتم عمليات التحليل ومن ثم المخرجات عن طريق السماعات و المحاور على شاشة المستخدم. من بعض تطبيقات هذا البرنامج في مجال الطب: عملية فصل الصوت إلى مجهور ومهموس تستخدم من قبل أطباء عيوب النطق لتقييم صوت المريض. بعض

الناس يصدر الصوت المجهور كصوت مهموس أو العكس. المريض يمكن أن يسجل صوته. ومن ثم أخصائي عيوب النطق يستطيع معرفة الصوت الذي في الأصل مجهور ثم صدر مهموسا أو العكس من خلال الألوان التي تبين الصوت المجهور من المهموس. بعض المرضى لديهم عيوب صوتية لها علاقة بالبتش ( Pitch frequency ). إذا كان المريض بالغاً ولا يزال ينتج بتش مثل التي لدى الأطفال. أخصائي عيوب النطق يستطيع معرفة البتش التي يولدها المريض من خلال البرنامج ومن ثم يتم تدريب المريض للحصول على البتش الصحيحة من خلال رؤية النتيجة على شاشة الحاسب الآلي.

الباب الخامس يتحدث عن كيفية استخدام برنامج تحليل الكلام بالتفصيل، من خلال شرح كيفية التسجيل ثم فتح الملف، ومن ثم شرح كل قائمة وأثرها ونتائجها وكذلك القوائم الفرعية مدعماً ذلك بالصور.

# DEVELOPMENT OF A SPEECH ANALYSIS PROGRAM USING MATLAB

**By**

**Dafer Saeed AL-Amri**

**A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering (Electronics and Communications)**

FACULTY OF ENGINEERING
KING ABDULAZIZ UNIVERSITY
JEDDAH – SAUDI ARABIA

DHUL-QADAH 1422 H / JANUARY 2002 G

# DEVELOPMENT OF A SPEECH ANALYSIS PROGRAM USING MATLAB

**By**

**Dafer Saeed AL-Amri**

**I certify that I have read this thesis and that in my opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science.**

**Thesis Supervisor**

---------------------------------------

**Dr. Ahmed S. Balamash**

# DEVELOPMENT OF A SPEECH ANALYSIS PROGRAM USING MATLAB

**By**

**Dafer Saeed AL-Amri**

**This thesis has been approved and accepted in partial fulfillment of the requirements for the degree of Master of Science.**

**Examiners:**

---------------------------------------------

**Dr. Ahmed S. Balamesh** **, Examiner/Supervisor**

---------------------------------------------

**Dr. Rabah W. Aldhaheri** **, Examiner**

---------------------------------------------

**Dr. Ali M. Rushdi** **, Examiner**

**Dedicated to**

*my dear father, mother, wife, and brothers with great love and appreciation*

## ACKNOWLEDGMENTS

**By the help of ALLAH, I prepared this thesis for the Master of Science degree in Electrical and Computer Engineering (Electronics and Communications).**

**I am grateful to my supervisor Dr. Ahmed Said Balamesh for this encouragement and all the time he has given to supervise my work. Many thanks are extended to the other members of the examining committee.**

# DEVELOPMENT OF A SPEECH ANALYSIS PROGRAM USING MATLAB

**DAFER SAEED AL-AMRI**

**ABSTRACT**

In this thesis, a complete speech processing and analysis software is generated. This package is based on MATLAB, and has the following elements of analysis: Energy, Spectrogram, Pitch, Formants, Voiced/Unvoiced classification, and Spectrum. Each of these types of analysis has its own options and parameters to change and control the analyzed operation. In this thesis, I deal with speech production, speech processing and analysis algorithms and mathematical background, and include some hints on graphical user interface design and a user manual for our speech analysis software.

# TABLE OF CONTENTS

CHAPTER III SPEECH PROCESSING



CHAPTER IV GRAPHICAL USER INTERFACE DESIGN

CHAPTER V SPEECH ANALYSIS SOFTWARE

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS

| | |
|---|---|
| ST | Short Time |
| STFT | Short Time Fourier Transform |
| LP | Linear Prediction |
| SLP | Speech Language Pathologies |
| rms | Root Mean Square |

# CHAPTER I

# INTORDUCTION

Speech processing and analysis relies on basic research in the speech and hearing sciences, some of which is centuries old, and much of which is ongoing. Speech analysis has very wide medical applications. It is begin used by medical doctors to diagnose some speech pathologies. Also, more recently some scholars of Quran are considering the use of speech analysis as a tool for teaching the art of perfect recitation (tajweed) [1].

In the production of speech, sound is generated in the vocal system either by vibration of the vocal cords or by the creation of turbulent air flow at a constriction. The sounds thus produced are spectrally shaped by the transmission characteristics of the vocal tract which consists of the pharynx and the oral cavity and, in some cases, the nasal tract. This speech production process can be modeled as a slowly varying source which produces a quasi-periodic pulse signal for voiced speech or a flat spectrum random signal for unvoiced speech. [2],[3]

Speech analysis is simply the process of estimation the time-varying parameters of the model for speech production from a speech signal that is assumed to be the output of the assumed model.

Speech analysis is a fundamental ingredient of almost all the important technical problems of speech communication. A speech analysis system when coupled to a corresponding speech synthesis system comprises a vocoder. Such systems can provide a means of efficient transmission or storage of speech signals. Speech analysis systems usually serve as the “front end” for systems for automatic speech recognition and for automatic speaker verification systems. Speech analysis is also involved in many aids to the handicapped such as speeded speech for the blind and visual training aids for teaching the deaf to speak. Still another area where speech analysis plays a fundamental role is in enhancing the quality of speech signals that have been degraded by noise, reverberation, or by production in an unusual atmosphere.

One of the most important parameters of speech is the fundamental frequency (pitch). A pitch detector is an essential component in a variety of speech processing systems. Besides providing valuable insights into the nature of the excitation source for speech production, the pitch contour of an utterance is useful for recognizing speakers, for speech instruction to the hearing impaired, and is required in almost all speech analysis-synthesis systems. A wide variety of algorithms for pitch detection have been proposed in the literature [4].

Pitch detection algorithms can be carried out in the time-domain [5], [6], [7], the frequency-domain [8], [9], or using a hybrid of both [10], [11].

Accurate determination of the resonant frequencies of the vocal tract (i.e., formants) in various articulatory configurations is of interest both in the synthesis and in the analysis of speech. For vowel sounds, in particular, the formant frequencies (especially the first and second formants) play a dominant role in determining which vowel is produced by a speaker and which vowel is perceived by a listener.

Several methods have been used to estimate the formant frequencies of vowels. Almost all of these techniques have as a common starting point; the transformation of the acoustic data into spectral form [12], [13], [14].

An algorithm for the determination of first three resonance frequencies (formants) of the speech signal has been proposed in [2].

It is frequently necessary to know whether a segment of speech under analysis is voiced or not. Many algorithms for voicing decision has been proposed in [15], [16]. In addition to the voicing decision, algorithms for silence [17], [18] and mixed [19] excitation decision have been developed.

One of the most powerful speech analysis technigues is the method of linear predictive analysis. This method has become the predominant technique for estimation the basic speech parameters, such as, pitch [20], formants [2].

In this thesis, we develop a speech analysis package using MATLAB, that performs a wide variety of speech analysis. This package is very flexible in the sense that adding new menus or new types of speech analysis is very easy and involves minimal changes.

In Chapters 2 and 3, we give a general background on speech production and analysis. Chapter 4 summarizes the general considerations taken into account in the design of the graphical user interface (GUI) used for our package. Chapter 5 is a

user's manual for our package. It gives detailed description of every menu and the parameters involved in each type of analysis. In Chapter 6, we conclude by summarizing the contributions of this thesis. We also list several suggestions for the future improvement of our speech analysis package.

# CHAPTER II
# SPEECH PRODUCTION

## 2.1 Introduction

Speech production represents many marvels and astonishing well-synchronized mental and neurological abilities and activities. It depends on a physical system consisting of various elements that may have dual functions such as the lungs (breathing and speech generation). The functional significance of the various elements pertinent to speech production varies from one language to another, depending on the acoustical sounds the language uses. For example, in Arabic, we use deep-throat sounds such as /ع، ق/, which are not present in languages like Chinese or English [21]. These deep-throat sounds rely heavily on certain articulators, which make their significance more important than in other languages that do not make such sounds. Moreover, we notice a degradation in sound quality in case of an imperfect articulator has a problem such as a congested nose or a broken front teeth.

## 2.2 Components of the Human Speech Production System

The speech waveform is an acoustic sound pressure wave that originates from voluntary movements of anatomical structures which make up the human speech production system. Figure 2.1 shows the components of the human speech production system. The gross components of the system are the *lungs*, *trachea*, *larynx*, *pharyngeal cavity*, *oral cavity* (mouth), and *nasal cavity* (nose).

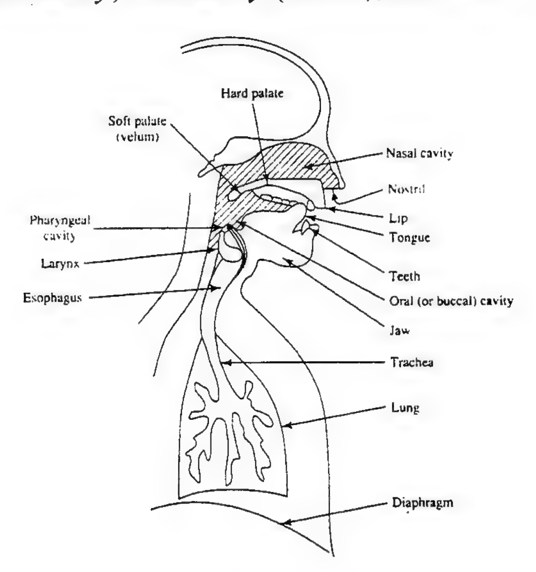



Figure 2.1 Human speech system

In the following we give a brief description of each component:

***Lungs*** (الرئتان) are considered the voice generator since the expelled air out of the lungs is used to form the various acoustical sounds by passing through the vocal and nasal tracts.

***Trachea*** (القصبة الهوائية) forms an air tunnel from the lungs to the throat.

***Larynx*** (الحنجرة) is the opening between the vocal cords at the entrance of the oral and nasal cavities. Generally, the larynx has four positions which are open (unvoiced sounds), open with repetitive closure (providing periodic excitation for voiced sounds), narrow opening (noisy sounds) such as /هـ / or closed such as / ء/.

***Vocal cords*** (الحبال الصوتية) are two strips of muscular fiber that meet at the top of the trachea. They are more rigid and longer in males than they are in females. They vibrate at 100-150 Hz in males and at 200-300 Hz in females. If the vocal cords vibrate with the acoustical sound, the sound is voiced such as in / ز، ذ / and if they don't, the sound will be unvoiced as in / س، ث/ [22].

***Pharyngeal cavity*** (التجويف البلعومي) forms a resonance cavity that amplifies some sounds such as / ظ، ض، ط، ص / and is responsible of producing the throat sounds such as / ق، ح، ع/.

***Oral cavity*** (التجويف الفمي) consists of the following articulators:

- ***Tongue*** is a moving articulator that takes different positions to control the air stream producing various sounds. As a sound producing element, the tongue can be viewed to consist of five parts. These are the tip producing sounds such as in /ن، ت، ث، ر/, the front producing sounds such as in /ي/, the middle producing sounds such as in /ص/, the back producing sounds such as in / ك/, and the root producing sounds such as in /ع/.
- ***Lips*** consist of a fixed part (upper lip) and a moving part (lower lip) that moves with the lower jaw. The upper and lower lips are equally involved in producing sounds like /م، ب، و / while only the lower lip has a distinct role in producing the sound /ف/ by touching the upper teeth.
- ***Teeth***: The front upper and lower teeth work with the tip of the tongue and the lower lip to produce sounds such as /ث، ذ، ت/, and work with the lower lip to produce the sound /ف/.
- ***Hard palate*** (الحنك الصلب) works with the tongue to produce sounds such as /ن، ل، ج/.

- ***Soft palate*** (الحنك الرخو) is a soft muscular tissue that include the velum (اللهاة). The velum is a hanged piece of meat that works with the tongue to produce sounds like /غ ،خ ،ك/.
- ***Epiglottis*** (لسان المزمار) is located at the glottis entrance. Beside blocking the breathing tunnel during eating or drinking, it helps in producing sounds like /ق/.
- ***Jaws*** (الفكان) consist of a moving (lower) part and a fixed (upper) part that work with the other articulators in producing various sounds.

***Nasal cavity*** (التجويف الأنفي) has a fixed shape that acts as a resonance cavity when air passes through it. It plays a major role in producing certain nasal sounds (الغنة) like in /م، ن/.

## 2.3 Speech Production

The acoustic sound pressure wave is generated through expelling inhaled air from lungs through trachea, causing the tensed vocal cords within the larynx to vibrate in the case of voiced sound. The airflow is chopped into quasi-periodic pulses that are frequency modulated when passing through the throat, mouth and nasal cavities. Repositioning the articulators consisting of vocal cords, velum, tongue, teeth, jaw, and lips produce different speech sounds.

One of the principal features of any speech sound is the manner of excitation. The two elemental excitation types are voiced and unvoiced excitation.

*Voiced* sounds are produced by forcing air through the glottis or opening between the vocal cords. The tension of the vocal cords is adjusted so that they vibrate in oscillatory fashion. The periodic interruption of the subglottal airflow results in quasi-periodic puffs of air that excite the vocal tract. The sound produced by the larynx is called voice. An example of a voiced sound is the vowel /الكسرة الطويلة/ in the utterance of "سبيل "

Unvoiced sounds are generated by forming a constriction at some point along the vocal tract, and forcing air through the constriction to produce turbulence. An example is the /س/ sound in "سبيل".

## 2.4 Human Speech System Model

Speech production can be modeled as an acoustic filtering operation. The acoustic filter consists of three main cavities; pharyngeal cavity, oral cavity, and nasal cavity. The pharyngeal and oral cavities are grouped into one unit referred to as the vocal tract. Figure 2.2 shows a speech production model with the vocal and nasal tracts shown as concatenations of tubes of non-uniform cross-sectional area. The sound wave propagation can be analyzed using the theory of wave propagation through transmission lines [22].

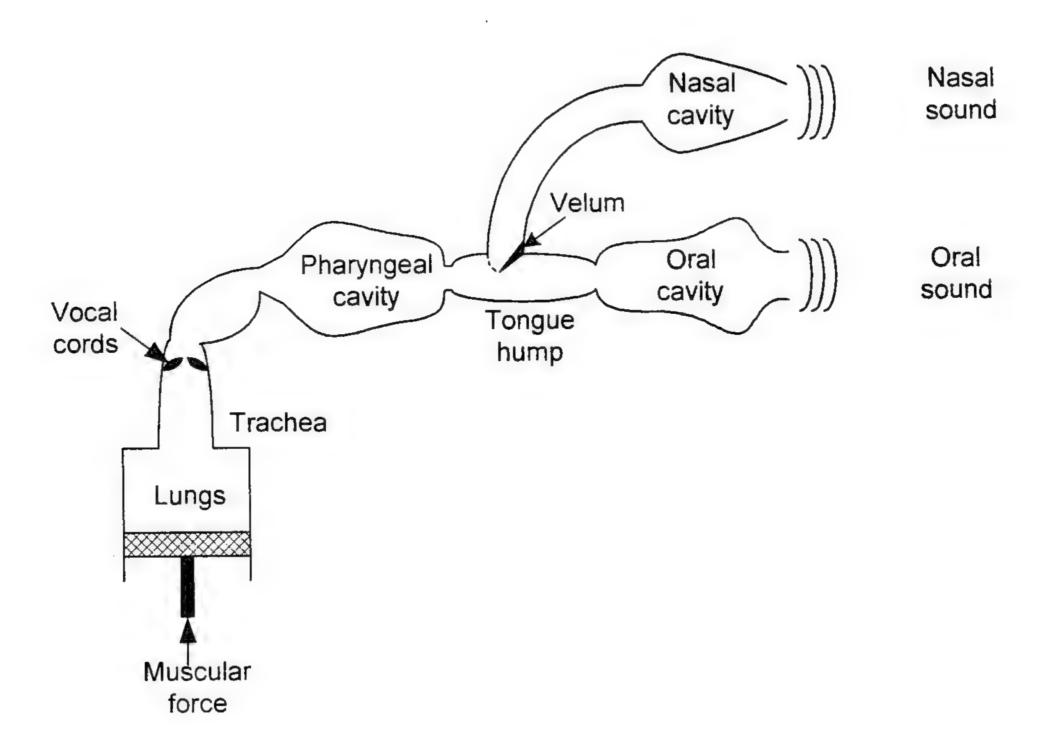



Figure 2.2 Human speech system model

The vocal tract begins at the output of the larynx and terminates at the lips. The average vocal tract length is about 17 cm in adult males, 14 cm in adult females and 10 cm in children. Repositioning of the vocal tract articulators causes the cross-sectional area of the vocal tract to vary along its length from zero to 20 $cm^2$. The nasal tract begins with the velum and ends at the nostrils. Typical length of the nasal tract in an adult male is 12 cm.

Acoustic coupling between the nasal and vocal tracts is controlled by the size of the opening at the velum. If the velum is lowered, the nasal tract is acoustically coupled to produce the nasal sounds of speech. For the production of non-nasal sounds, the velum is drawn up tightly toward the back of the pharyngeal cavity,

effectively sealing off the entrance to the nasal cavity and decoupling it from the speech production system.

## 2.5 Discrete Time Model for Speech Production

Voiced excitation can be modeled as an impulse train generator producing a sequence of unit impulses which are spaced by the desired fundamental period. This signal, in turn, excites a linear system whose impulse response $g(n)$ has the desired glottal wave shape. Unvoiced excitation can be modeled using white noise. By switching between the voiced and unvoiced excitation generator we can model the changing mode of excitation. Vocal tract is modeled be the transfer function $H(z)$ as shown in Figure 2.3 [22].

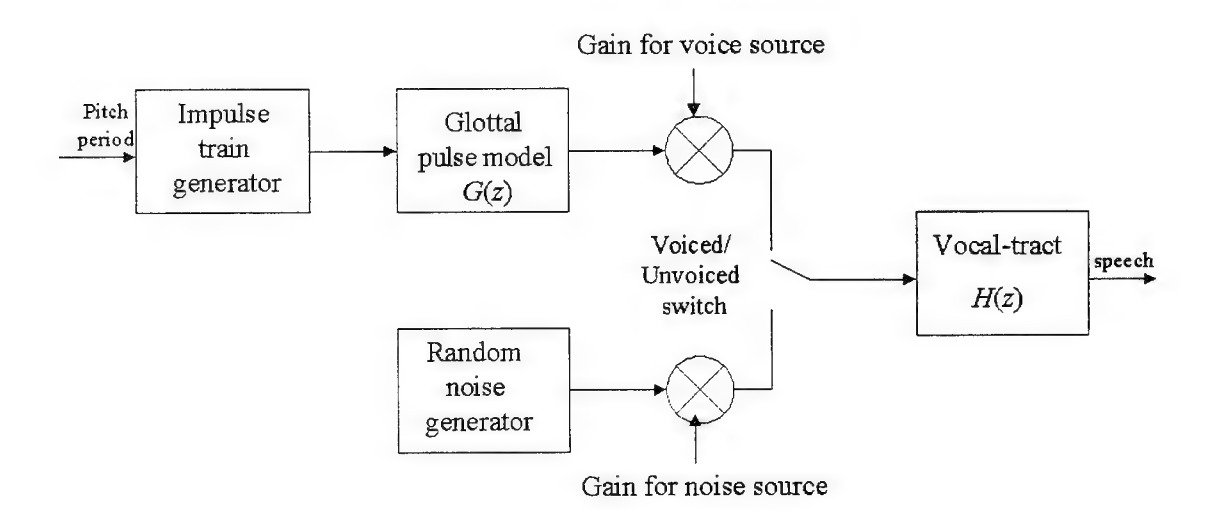



Figure 2.3 Discrete time model for speech production

## 2.6 Acoustic Characteristics of Sounds

In all languages, sound can be classified based on various descriptive features like manner and place of articulation, articulator organs, type of excitation, sound's volume, length, resonance, and many other features. Acoustical sounds can be covered within three broad classes of sounds. They are vowels, semivowels, and consonants. Each of these classes may be further broken down into sub-classes. They are related to the manner, and place of articulation of the sound within the vocal tract. Figure 2.4 shows a classification of the Arabic language phonemes [30].

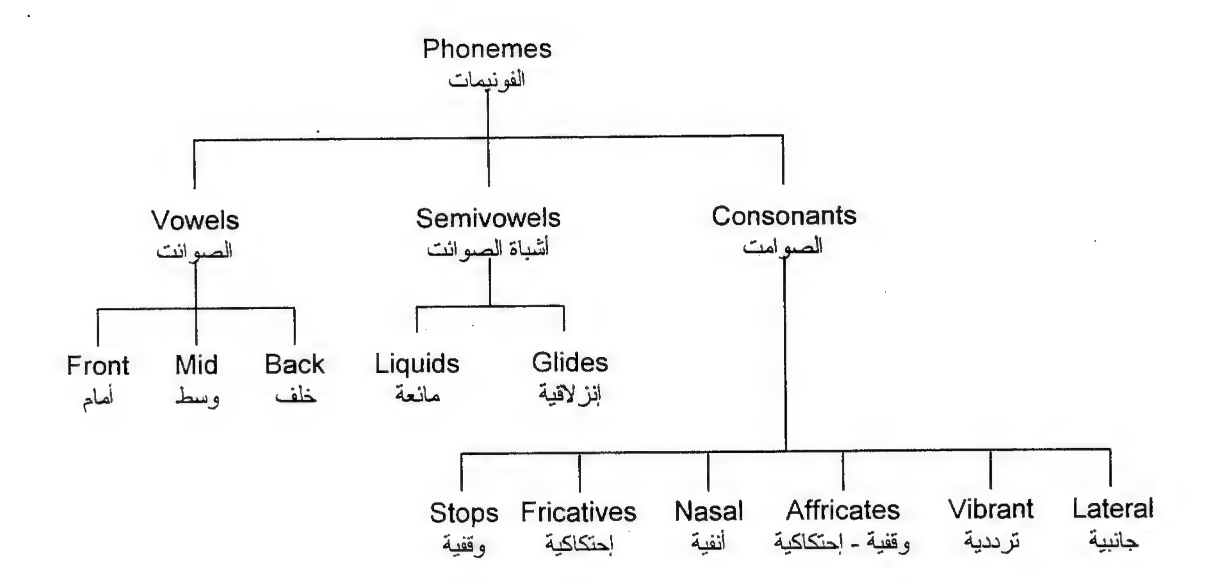



Figure 2.4 Classification of phonemes

### 2.6.1 Vowels

Vowels are voiced sounds produced by exciting a fixed vocal tract with quasi-periodic pulses of air caused by vibration of the vocal cords. Vowels are long in duration (compared to consonant sounds) and are spectrally well-defined. In vowels, the expelled air does not meet any obstacle in its way out. Vowels constitute 48 % of the Arabic language. In Arabic language there are six vowels:

(الفتحة القصيرة، الكسرة القصيرة، الضمة القصيرة، الألف- فتحة طويلة، الياء- كسرة طويلة، الواو- ضمة طويلة).

Vowel sounds are characterized primarily by the hump and height position of the tongue. Figure 2.5 shows the tongue positions, waveforms, and formant (resonant frequencies of the vocal tract) plots for the vowels /a/ and /i/.

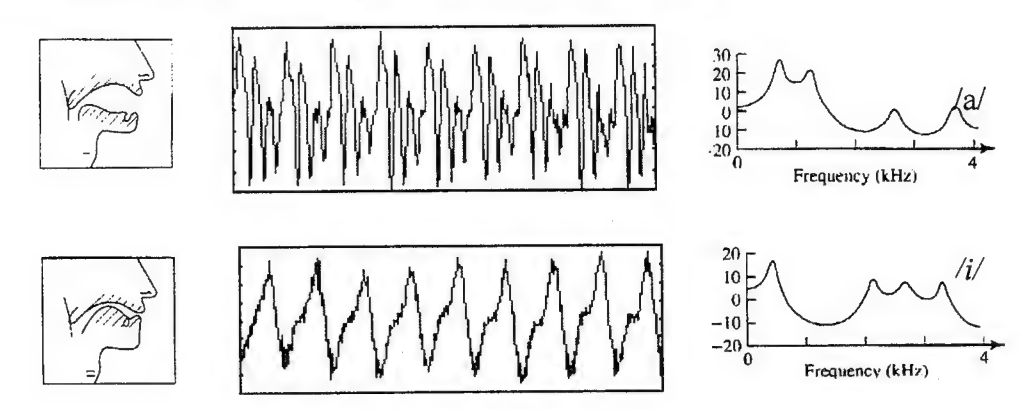



Figure 2.5 Vocal tract, waveform, and formants of sounds /a/ and /i/

In forming the vowel /a/ (فتحة طويلة) as in ”جمال“ the vocal tract is open at the front and somewhat constricted at the back by the main body of the tongue. In contrast, the vowel /i/ (كسرة طويلة) as in ”كيس“ is formed by raising the tongue toward the palate, thus causing a constriction at the front and increasing the opening at the back of the vocal tract. The acoustic waveforms in Figure 1.5 show the quasi-periodic characteristic of voiced sounds. The vowel (كسرة طويلة) shows a low frequency damped oscillation upon which a relatively strong high frequency oscillation is superimposed. This is consistent with a low first formant and high second and third formants. Table 2.1 shows the first, second, and third formant frequencies of some American English vowels [23].

| Symbol | Example | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|---|
| /i/ | Beet | 270 | 2290 | 3010 |
| /I/ | Bit | 390 | 1990 | 2550 |
| /a/ | Hot | 730 | 1090 | 2440 |
| /U/ | Foot | 440 | 1020 | 2240 |
| /u/ | Boot | 300 | 870 | 2240 |

Table 2.1 Formant frequency for typical vowels [23].

The spectrograms for the two sounds are shown in Figure 2.6. Where the spectrogram is a two dimensional representation of the time dependent spectrum in which the vertical dimension represents frequency and the horizontal dimension represents time. As depicted, vowels are easily identified due to their high energy resulting in intense formant patterns.




Figure 2.6 Spectrum of the /a/ and /i/ sounds.

The overall length of the pharyngeal-oral tract, the location of constrictions along the tract, and the narrowness of the constrictions affect formant frequency locations for vowels. Vowel formants are closely related to the hump and height position of the tongue. Figure 2.7 shows the second formant ($F_2$) drawn on the y-axis versus the first formant ($F_1$) drawn on the x-axis. The three vowels /i/, /a/, /u/ accounted for in many languages including Arabic are distinctively classified from each other at the vertex of the triangle formed by connecting their respective extreme formant locations as shown in the figure [23], [24].

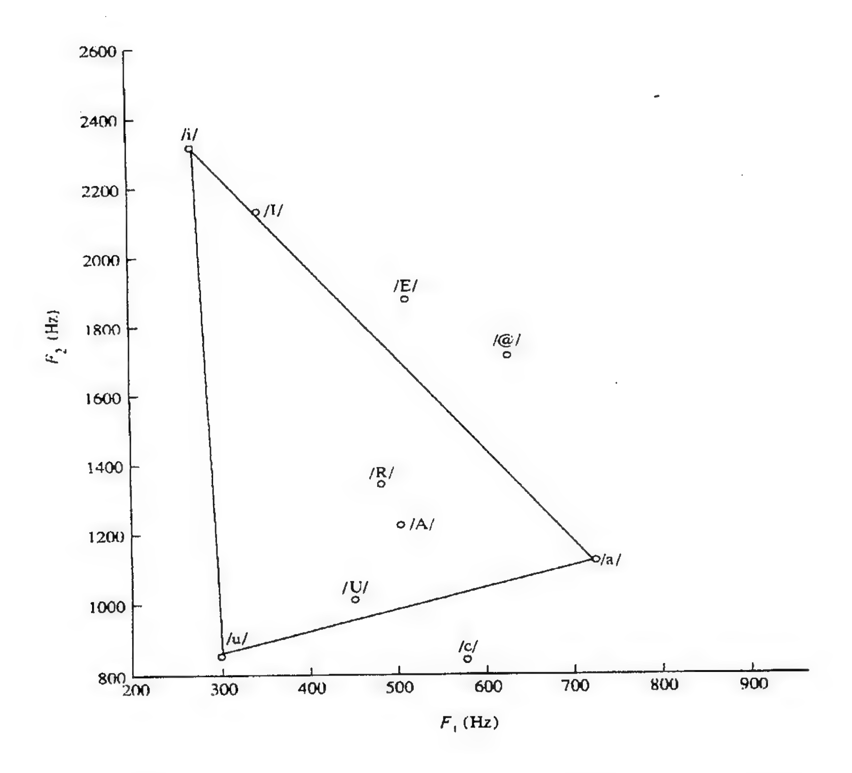



Figure 2.7 The vowel triangle plot of F1 and F2 for vowels in English [22].

As depicted in Figure 2.7, the greater the constriction in the front half of the vocal tract, the lower the first formant. Thus /i/, which has a great amount of constriction in the front half of the vocal tract has a low first formant while /a/, which is realized with a much wider opening in the front half of the vocal tract has a high first formant. The greater the constriction in the pharynx region, the higher the first formant. Thus /o/ has a lower first formant than /i/. The second formant is lowered by back tongue constriction while it is raised by front tongue constriction. Thus /u/ has a low second formant, while /i/ has a high second formant. Lip rounding lowers the frequencies of higher formants (especially F2 and F3) [24].

### 2.6.2 Consonants

Generally speaking, consonants have weaker energy than vowels. The first formant generally rises during the transition from a consonant to a vowel (and inversely falls from a vowel to a consonant) especially if the consonant is voiced. In contrast with vowels, consonants normally have specific articulators and place of articulation. Their number is much more than the vowels in all languages, and they can be voiced and unvoiced. Consonants are classified as stops, fricatives, nasals, affricates, repetitive, and sideways. The various classes of consonants are discussed below:

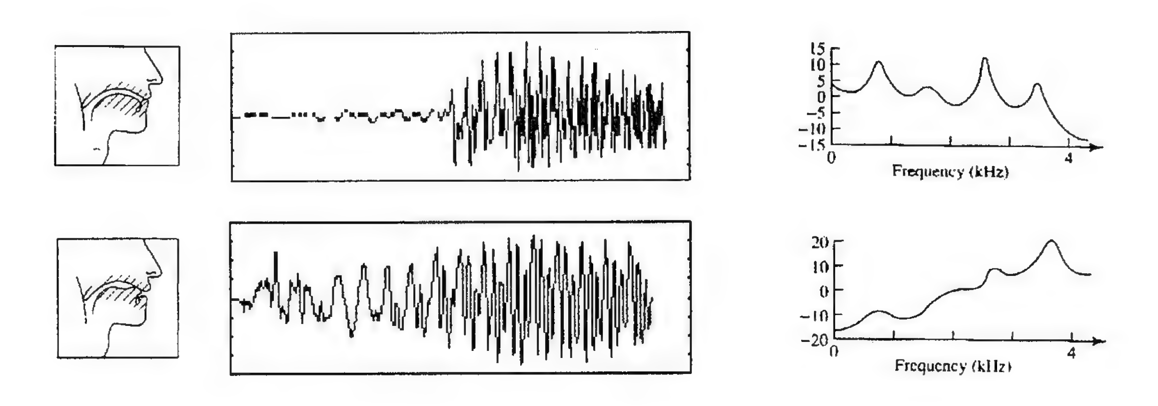



Figure 2.8 Articulatory position, waveform, and formants of sounds /ب/ and /ت/ [22].

***Stops (Plosives):*** Stops are characterized by a silent period during the consonantal closure where pressure is built up followed by an abrupt onset of energy – a noise burst- at consonantal release. There are eight stop sounds in Arabic, which are /ت، د، ط، ض، ك، ق، ء، ب/ .The airflow stops at different places in the oral tract. For example in the /ب/ sound the airflow stops at the lips and in the /ت، د/ sounds stop at the tongue tip touching the teeth. The properties of stop sounds are highly influenced by the vowel that follows the stop consonant. Figure 2.8 shows articulatory position, waveforms and formants of the voiceless stop consonants /ب/ and /ت/.

It is readily seen that the duration and frequency content of the friction noise and aspiration varies greatly with the stop consonant. Although during the period of total constriction in the tract there is no sound radiated from the lips, there is often a small amount of low frequency energy radiated through the walls of the throat. This occurs when the vocal cords are able to vibrate even though the vocal tract is closed

at some point. The vibration of the vocal cords distinguishes voiced from unvoiced stops.

***Fricatives:*** The narrow constrictions associated with fricatives result in the production of turbulent sounds. There are thirteen fricative sounds in Arabic, which are / ف، ث، س، ص، ش، خ، ح، هـ، ذ، ز، ظ، غ، ع /. Because of their relatively low energy, some fricatives such as /ف/ is difficult to identify. On the other hand, whistling fricatives /س، ص، ز، ش/ can be identified from the spectral shape sampled during the fricative itself. The location of the constriction determines which fricative sound is produced. For example, for the fricative /ف/ the constriction is near the lips; for /ذ/ it is near the teeth; for /س/ it is near the middle of the oral tract; and for /ش/ it is near the back of the oral tract. Figure 2.9 shows the waveforms and spectrograms of the fricatives /ف/ and /ش/. The non-periodic nature of fricative excitation is obvious in the waveform plots.

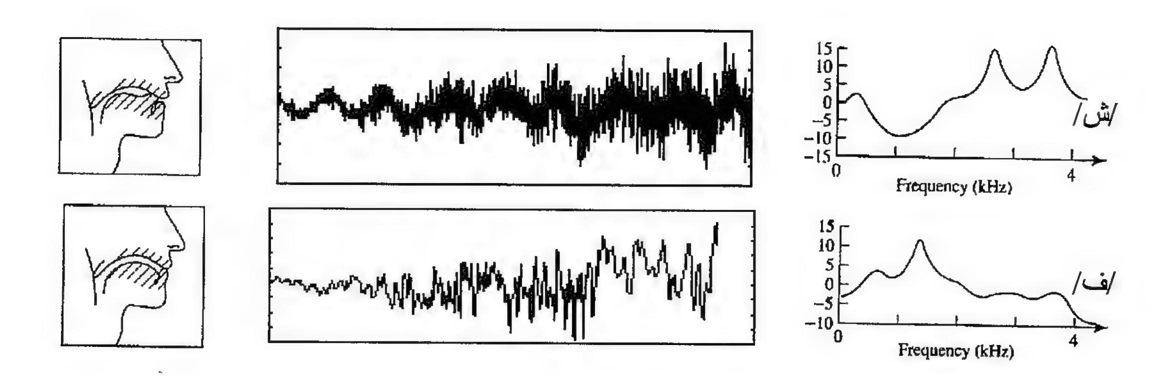



Figure 2.9 Articulatory position, waveform, and formant for sounds /ش/ and /ف/ [22].

The voiced fricatives as in /ز/ differ from the unvoiced ones as in /س/ . For voiced fricatives the vocal cords are vibrating, and thus one excitation source is at the glottis. However, since the vocal tract is constricted at some point forward of the glottis, the airflow becomes turbulent in the neighborhood of the constriction.

***Nasals:*** Nasal consonants are characterized by letting air go through the nasal cavity and closing of the oral cavity at some point. In nasal sounds the velum is lowered so the sound radiates at the nostrils. There are two nasal sounds in Arabic, which are /م، ن/. The constriction for the nasal consonants /م/ is at the lips and for /ن/ is just back of the teeth. Although the oral cavity is constricted, it is still

acoustically coupled to the pharynx. Thus, the mouth serves as a resonant cavity that traps acoustic energy at certain places in the oral cavity [25].

Figure 2.10 shows typical speech waveforms and spectrograms for two nasal consonants. The spectrograms show a concentration of low frequency energy with a mid-range of frequencies that contains no prominent peaks. This is because of the particular combination of resonances and anti-resonances that result from the coupling of the nasal and oral tracts.

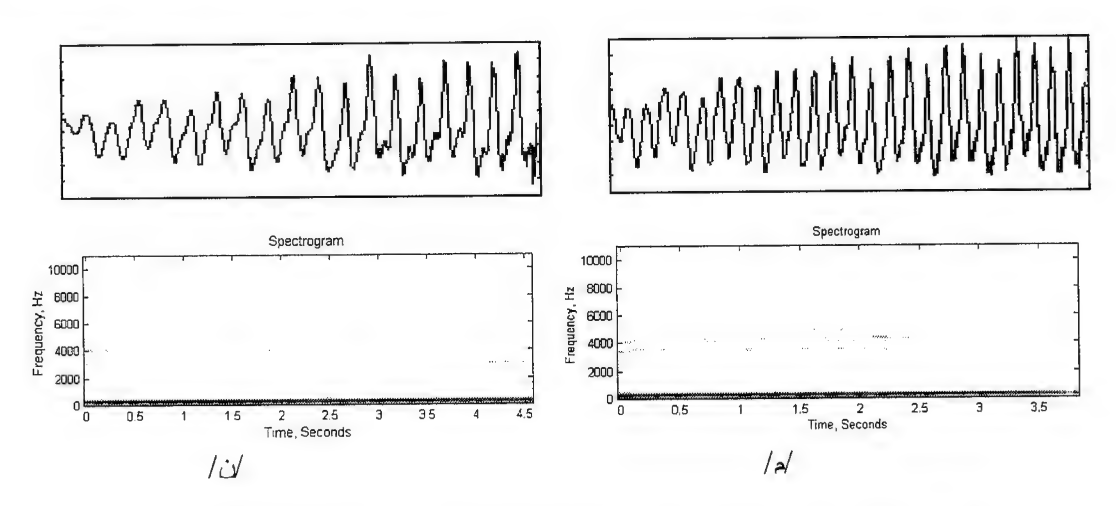



Figure 2.10 Waveform and spectrogram of nasal sounds /ن/&/م/ [22].

***Affricates:*** The Affricates consonants are dynamical sound that can be modeled as a concatenation of plosive and fricative. There is one affricate sound in Arabic which is /ج / .Voiced affricates /ج/ can be modeled as the concatenation of the stop /د/ and the fricative /ژ/ [25].

***Vibrant:*** The repetitive sounds in Arabic is the /ر/ which may repeat itself 2-5 times when used in speech.

***Lateral sounds:*** Lateral sound is the sound that emerges from sides of the mouth. The sideways sound in Arabic is /ل / .

### 2.6.3 Semivowels

The consonants and vowels are intermixed sometime to provide semi-consonant and semi-vowel sounds. The group of sounds consisting of /و/ and /ي/ are characterized as semivowels because of their vowel-like nature. They are generally characterized by a gliding transition in vocal tract area function between adjacent phonemes. Thus the acoustic characteristics of these sounds are strongly influenced by the context in which they occur. An example of the semivowel /و/ is shown in Figure 2.11 [25].

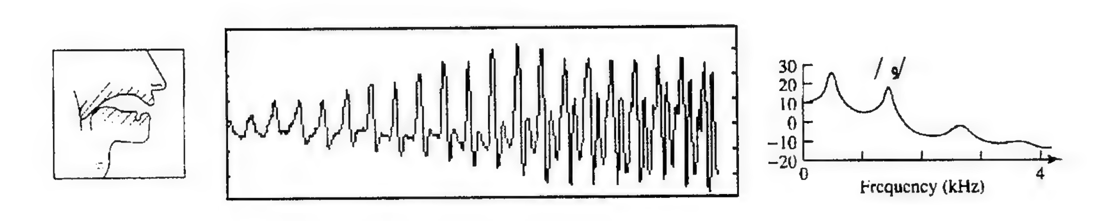



Figure 2.11 Articulatory position, waveform, and formant for semivowel /و/ [22].

## 2.7 Phonology

Speech signals are sequences of sounds arranged and governed by rules associated with language. The scientific study of the language and the manner in which these rules are used in human communication is referred to as linguistics. The science that studies the characteristics of human sound production, especially for the description, classifications, function, and transcription of speech, is called phonetics.

The phoneme is defined as the smallest possible analytical (functional and configurable) sound unit capable of distinguishing one utterance from another. In each language the phonemes are tied together to form a group configuration that has common relations and identified structure. This configuration organizes sound-segments relation, location, accent, frequency, interaction, flow, meaning, homogeneity, contradiction, and indication etc.

A phoneme, by itself, is independent of its location in an utterance and the other phonemes surrounding it. However, each phoneme has several ways (allophones) to pronounce it depending on its location in speech.

A phoneme can be classified as either a continuant, or a non-continuant sound. Continuant sounds are produced by a fixed (non-time-varying) vocal tract configuration excited by the appropriate source. The class of continuant sounds includes the vowels, the fricatives (both unvoiced and voiced), and the nasals. The

remaining sounds (diphthongs, semivowels, stops and affricates) are produced by a changing vocal tract configuration. These are therefore classified as non-continuants [26].

# CHAPTER III
# SPEECH PROCESSING AND ANALYSIS

## 3.1 Introduction

Speech processing and analysis can be done either in time domain or in frequency domain. In time domain processing, we deal with the waveform of the speech signal directly for estimating important features. In frequency domain, we deal with the Foureir representation of the speech signal. Fourier representation often serves to place in evidence certain properties of the signal that may be obscure or at least less evident in the original signal.

## 3.2 Time Domain Processing and Analysis of Speech

Our goal in processing the speech signal is to obtain a more convenient or more useful representation of the information carried by the speech signal. The required precision of this representation is dictated by the particular information in the speech signal that is to be preserved or, in some cases, made more prominent. For example, the purpose of the digital processing may be to facilitate the determination of whether a particular waveform corresponds to speech or not. In a similar but somewhat more complicated vein, we may wish to make a 2-way classification as to whether a section of the signal is voiced speech, or unvoiced speech. In such cases, a representation which discards "irrelevant" information and places the desired features clearly in evidence is to be preferred over a more detailed representation that retains all the inherent information.

A sequence of samples (11025 samples/sec) representing a typical speech signal (بسم الله الرحمن الرحيم) is shown in Figure 3.1. It is evident form this figure that the properties of the speech signal change with time.

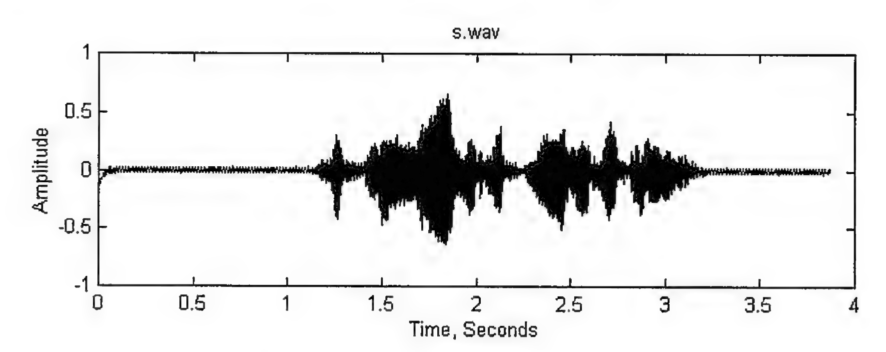



Figure 3.1 (بسم الله الرحمن الرحيم) Speech waveform

For example, the excitation changes between voiced and unvoiced speech, there is significant variation in the peak amplitude of the signal, and there is considerable variation of the fundamental frequency within voiced regions when the time scale is maximized. The fact that these variations are so evident in a waveform plot suggests that simple time domain processing techniques should be capable of providing useful representations of such signal features as excitation mode, pitch, and possibly even vocal tract parameters such as formant frequencies.

The underlying assumption in most speech processing schemes is that the properties of the speech signal change relatively slowly with time. This assumption leads to a variety of "Short-Time" (ST) processing methods in which short segments of the speech signal are isolated and processed as if they were short segments from a sustained sound with fixed properties. This is repeated (usually periodically) as often as desired. Often these short segments, which are sometimes called analysis frames, overlap one another. The result of the processing on each frame may be either a single number, or set of numbers. Therefore, such processing produces a new short-time sequence which can serve as a representation of the speech signal.

Most of the ST processing techniques can be represented mathematically in the form

$$Q_n = \sum_{m=-\infty}^{\infty} T[x(m)]w(n-m) \qquad (3.1)$$

The speech signal $x(m)$ (possibly after linear filtering to isolate a desired frequency band) is subjected to a transformation, $T$ [ ], which may be either linear or nonlinear. The resulting sequence is then multiplied by a window sequence $w(n\text{-}m)$ positioned at a time corresponding to sample index $n$. The product is then summed over all nonzero values [22].

### 3.3 Short Time Energy

The amplitude of the speech signal varies appreciably with time. In particular, the amplitude of unvoiced segments is generally much lower than the amplitude of voiced segments. The short-time energy of the speech signal provides a convenient representation that reflects these amplitude variations. In general, we can define the short-time energy as

$$E_n = \sum_{m=-\infty}^{\infty}[x(m)w(n-m)]^2 \tag{3.2}$$

This expression can be written as

$$E_n = \sum_{m=-\infty}^{\infty} x^2(m)\cdot h(n-m) \tag{3.3}$$

where

$$h(n) = w^2(n) \tag{3.4}$$

Equation (3.3) show that the signal $x^2(n)$ is filtered by a linear filter with impulse response $h(n)$ as given by Equation (3.4).

If we use the rectangular window which is

$$h(n) = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

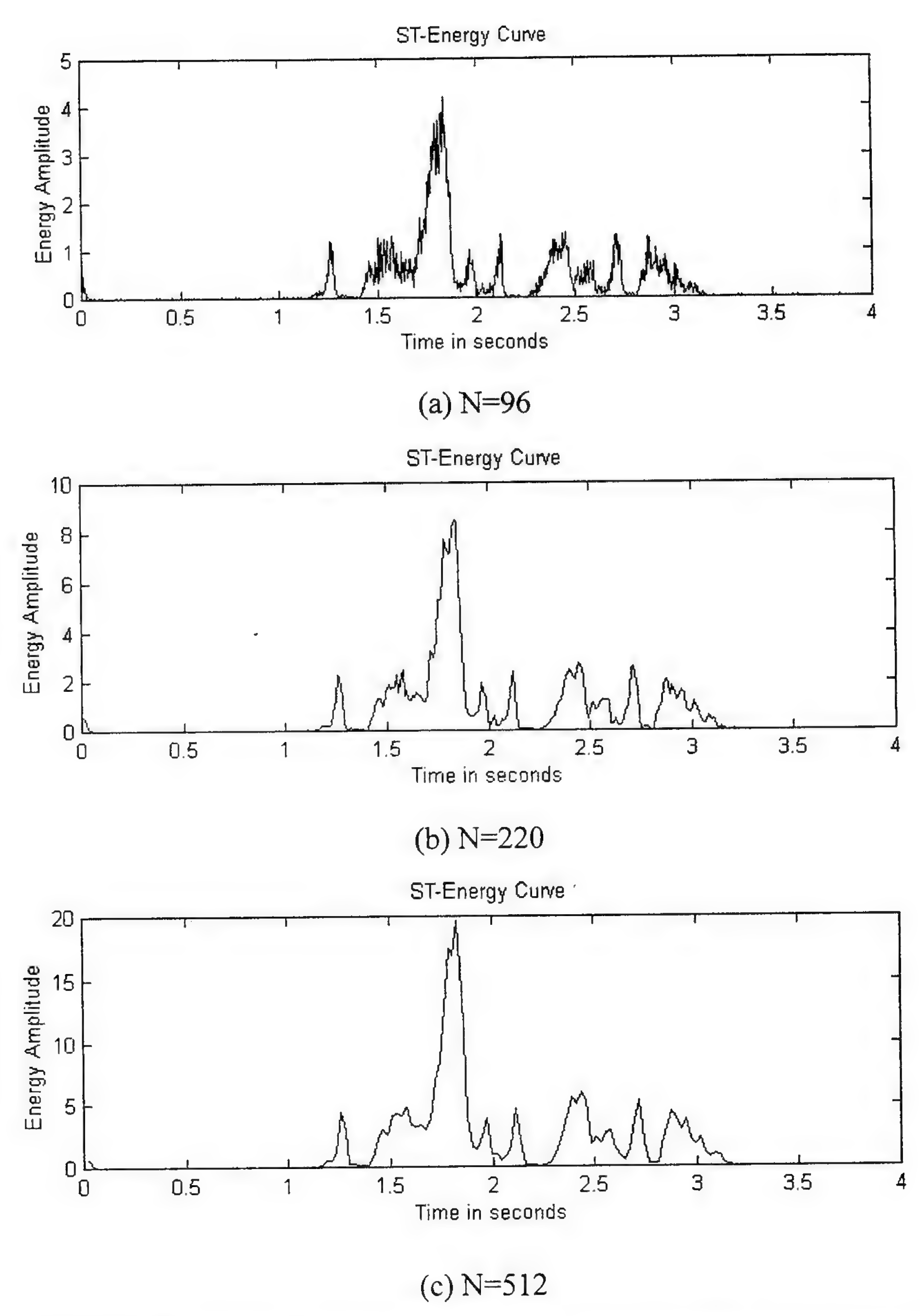
the Short Time energy will be

$$E_n = \sum_{m=n-N+1}^{n} x^2(m) \tag{3.6}$$

That is, the short time energy at sample n is simply the sum of the squares of the $N$ samples $n$-$N$+$1$ through $n$ [22].

Increasing the length $N$, simply decreases the bandwidth of the window. If N is too small, i.e., on the order of a pitch period or less, $E_n$ will fluctuate very rapidly, depending on the exact details of the waveform. If N is too large, i.e., on the order of several pitch periods, $E_n$ will change very slowly and thus will not adequately reflect the changing properties of the speech signal. Unfortunately, this implies that no single value of N is entirely satisfactory, because the duration of a pitch period varies form about 20 samples (at a 10 kHz sampling rate) for a high pitch female or a child, up to 250 samples for a very low pitch male. A suitable practical choice for N is on the order of 100-200 for a 10 kHz sampling rate (i.e., 10-20 ms duration).

Figure 3.2 shows the effects of varying the duration of the window on the energy computation for the utterance (بسم الله الرحمن الرحيم) spoken by a male speaker using a Hamming window. It is readily seen that as N increases, the energy becomes smoother.




(a) N=96

(b) N=220

(c) N=512

Figure 3.2 The effect of varying window length on energy computation

The major significance of $E_n$ is that it can help in distinguishing voiced speech segments from unvoiced speech segments. As can be seen in Figure 3.2, the values of $E_n$ for the unvoiced segments are significantly smaller than for voiced segments. The energy function can also be used to locate approximately the time at which voiced speech becomes unvoiced, and vice versa [22].

### 3.4 Short Time Autocorrelation Function

One of the most important of speech signal representations in terms of time domain processing is the autocorrelation function.

The autocorrelation function of a discrete-time deterministic signal *x*(*m*) is defined as

$$R(k) = \sum_{m=-\infty}^{\infty} x(m)x(m+k) \tag{3.7}$$

The autocorrelation function representation of the signal is a convenient way of displaying certain properties of the signal. If the signal is periodic with period *P* samples, then it is easily shown that

$$R(k) = R(k+P) \tag{3.8}$$

i.e., the autocorrelation function of a periodic signal is also periodic with the same period. Other important properties of the autocorrelation function are:

1. It is an even function, i.e., $R(k) = R(-k)$.
2. It attains its maximum value at $k = 0$,, i.e., $|R(k)| \leq R(0)$ for all k.
3. The quantity $R(0)$ is equal to the energy for deterministic signals or the average power for random or periodic signals.

Thus, the autocorrelation function contains the energy as a special case.

Even more important is the convenient way in which periodicity is displayed. If we consider Equation (3.8) together with properties (1) and (2), we see that for a periodic signal of period *P*, the autocorrelation function attains a maximum at samples $0, \pm P, \pm 2P, \ldots$. That is, regardless of the time origin of the signal, the period can be estimated by finding the location of the first maximum in the autocorrelation function. This properly makes the autocorrelation function an attractive basis for estimating periodicities in all sorts of signals, including speech.

The short-time autocorrelation function can be defined as

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)w(n-m)x(m+k)w(n-k-m) \tag{3.9}$$

This equation can be interpreted as follows: first a segment of speech is selected by multiplication by the window, then the deterministic autocorrelation definition Equation (3.1) is applied to the windowed segment of speech.

It is easily verified that

$$R_n(-k) = R_n(k) \tag{3.10}$$

If we define

$$h_k(n) = w(n)w(n+k) \tag{3.11}$$

Then Equation (3.9) can be written as

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)x(m-k)h_k(n-m) \tag{3.12}$$

Thus the value at time $n$ of the k$^{th}$ autocorrelation "lag" is obtained by filtering the sequence $x(n)x(n-k)$ with a filter with impulse response, $h_k(n)$ [22].

## 3.5 Voiced/Unvoiced Classification Using Autocorrelation Method

Voiced/Unvoiced classification is performed using autocorrelation function, through extracting some feature from this function. In the beginning speech is low-pass filtered with a 6-pole Butterworth filter having a cutoff frequency of 600 Hz as suggested by Gold and Rabiner [6]. The autocorrelation function is computed for each 51.2 ms using a rectangular window with 50% overlap.

Three features derived from the autocorrelation function are:

1. $e$, the rms energy of the segment.
2. $p$, the maximum value of the autocorrelation function over the pitch range normalized by the value at zero lag.
3. $r$, the rms of the normalized autocorrelation function over the pitch range.

These features are calculated as follows:

$$e = \left( \frac{R(0)}{segment\ length} \right)^{1/2} \tag{3.13}$$

$$p = \frac{R(K)}{R(0)} \tag{3.14}$$

$$r = \left[ \frac{1}{l_{\max} - l_{\min}} \sum_{l_{\min}}^{l_{\max}} \left( \frac{R(i)}{R(0)} \right)^2 \right]^{1/2} \tag{3.15}$$

where $K$ is the index of the maximum peak (largest value) in the pitch range, $l_{min}$ and $l_{max}$ are the min term and the max term for the pitch range, respectively [3].

### 3.6 Pitch Detection Using the Autocorrelation Method

In pitch detection using autocorrelation function, the segment under analysis must be a voiced segment, because the fundamental frequency (pitch) is valied only for voiced speech. In the beginning, a 51.2 ms voiced speech segment is low-pass filtered with a 6-pole Butterworth filter having a cutoff frequency of 600 Hz. Then the short-time autocorrelation function is calculated for this segment.

A peak picking algorithm is applied to the autocorrelation function of the segment. This algorithm starts by choosing the maximum peak (largest value) in the pitch range of 50 to 333 Hz (3 to 20 ms). This peak has a corresponding lag (*L*). The period corresponding to *L* is the first estimate of the pitch period.

As shown in Figure 3.3, the algorithm checks for peaks at one-half, one-third, one-fourth, one-fifth, and one-sixth of the first estimate of the pitch period. If *L*/2 (rounded up) is within the pitch range, the maximum value of the autocorrelation within (*L*/2) - 5 to (*L*/2) + 5 is located. If (*L*/2) - 5 is less than 3 ms, the lower limit of the pitch range is chosen instead of (*L*/2) – 5. If this new peak is greater than one-half of the old peak, the new corresponding lag replaces the old corresponding lag *L*. This new L might not be exactly *L*/2 as shown in Figure 3.3. We now have a new *L*, which presumably is corrected for the possibility of a pitch period doubling error. This test is performed again to check for double doubling error, (fourfold errors). If the most recent test fails, a similar test is performed for tripling errors of this new *L*. This test checks for pitch period errors of sixfold. If the original test failed, the original *L* is tested (in a similar manner) for tripling errors and errors of fivefold. With a sampling rate of $f_s$, the final value of *L* is used to calculate the pitch estimate *F* by [3]:

$$F = \frac{f_s}{L} \tag{3.16}$$

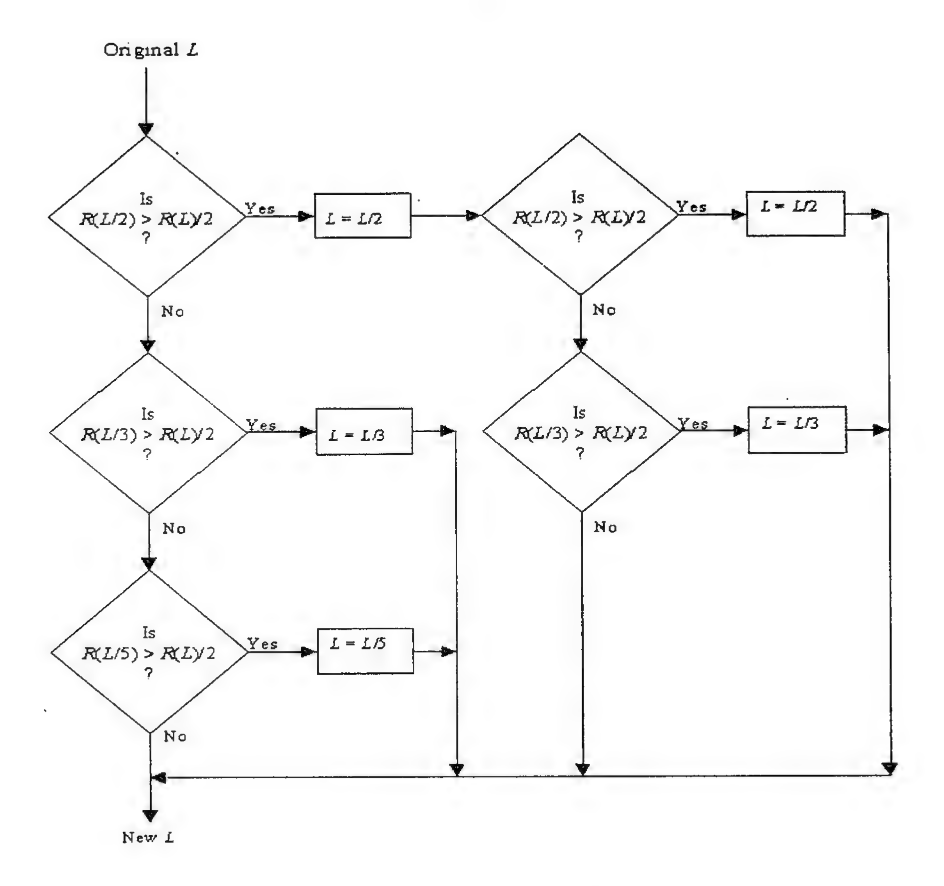



Figure 3.3 Decision process for the peak picking algorithm

### 3.7 Linear Prediction (LP) Analysis

One of the most important of types of time domain dependent processing is LP. In the following there is a brief exposition of LP.

We can approximate the sampled speech waveform $s(n)$ by another sequence $\hat{s}(n)$, by linearly predicting from the past $p$ samples of s(n) as follows

$$\hat{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \tag{3.17}$$

The unknowns $a_k$ in (3.17), can be determined by minimizing the mean squared error, $E$, between $s(n)$ and $\hat{s}(n)$ over N samples of $s(n)$:

$$E = \frac{1}{N}\sum_{n=0}^{N-1}[s(n) - \hat{s}(n)]^2 \tag{3.18}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1}[s(n) - \sum_{k=1}^{P} a_k s(n-k)]^2 \tag{3.19}$$

By setting $\partial E/\partial a_j$ to 0 for $j = 1,2,...,p$, and simplifying, one obtains:

$$\sum_{k=1}^{p} a_k \phi_{jk} = x_j \qquad j = 1,2,...p \tag{3.20}$$

where

$$\phi_{jk} = \sum_{n=0}^{N-1} s(n-j)s(n-k) \tag{3.21}$$

$$x_j = \phi_{j0} \tag{3.22}$$

Once the coefficients $a_k$ are available, it is an easy matter to obtain approximated properties of $s(n)$ from those of $\hat{s}(n)$. One simply evaluates the magnitude of the transfer function *H*(*z*) of the filter represented by the coefficients $a_k$, at *N* equally spaced samples along the unit circle in the z-plane:

$$H(z) = 1/(1 - \sum_{k=1}^{P} a_k z^{-k}) \tag{3.23}$$

where (3.23) is evaluated at $z = r \cdot \exp[j(2\pi n / N)]$ for $n = 0,1,..., N-1$, where $r$=1 [27]

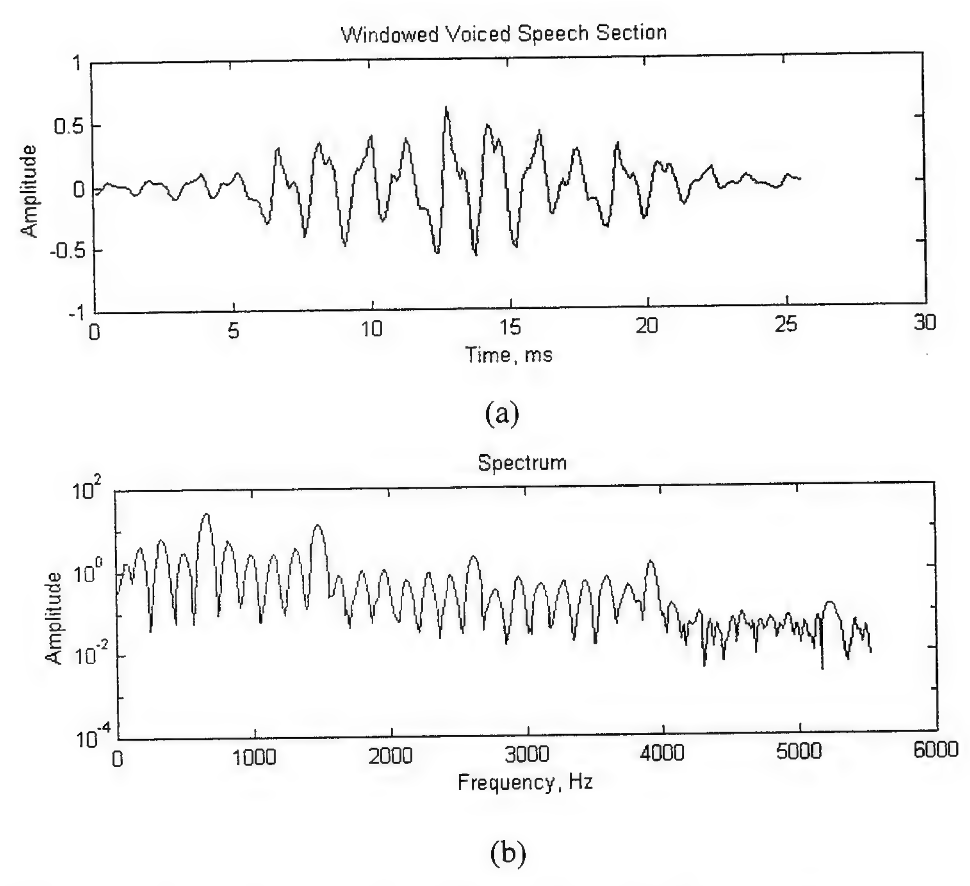
## 3.8 Frequency Domain Processing and Analysis of Speech

The spectral representation reflects the time-varying properties of the speech waveform. A useful definition of the Short-Time Fourier Transform (STFT) is

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} w(n-m)x(m)e^{-j\omega m} \tag{3.31}$$

The spectrum of speech would be the product of the frequency response of the vocal tract system and the spectrum of the excitation. Thus, it is to be expected that the spectrum of the output would reflect the properties of both the excitation and the vocal tract frequency response [22].

In Equation (3.31), *w*(*n-m*) is a real "window" sequence which determines the portion of the input signal that receives emphasis at a particular time index, *n*.




(b)

Figure 3.4 Spectrum analysis for voiced speech using a 25.6 ms Hamming window

Figure 3.4 (a) shows a 25.6 ms windowed section of a voiced speech. Figure 3.4 (b) shows the resulting log magnitude spectrum. The periodicity of the signal is clearly seen in part (a) (the time waveform) as well as in part (b) in which the fundamental frequency and its harmonics show up as narrow peaks at equally spaced frequencies in the STFT. Finally the spectrum shows a tendency to fall off at higher frequencies.

### 3.9 Smoothed Spectrum Using Linear Prediction

The variance of the STFT is usually large. Smoother estimate of the Power Spectral Density (PSD) an be obtained using parametric methods which assume certain models for the speech signal [22].

Using LP analysis, a smooth spectral density can be obtained as

$$P_x(\omega) = \frac{\sigma_w^{\,2}}{\left|1 + \sum_{l=1}^{p} a_l \, e^{-j\omega}\right|^2} \tag{3.25}$$

where $\sigma_w$ is the minimum mean squared error.

This will give a smooth spectrum that very well approximates the envelope of the spectrum obtained by STFT.

In Figure 3.5, the dashed line represents a smoothed spectral density of a 25.6 ms segment from the waveform in Figure 3.4(a). The smoothing is done using an LP model of order $p = 12$.

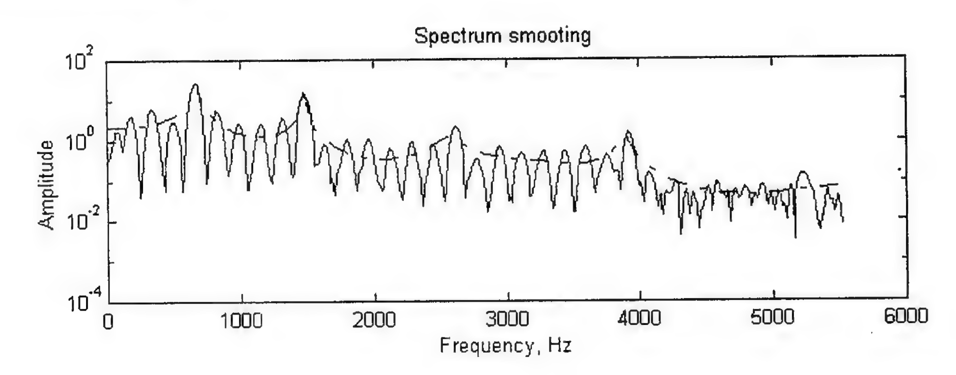



Figure 3.5 Smoothed spectrum

### 3.10 Spectrogram

One of the earliest embodiments of the ST Fourier representation was the sound spectrogram, a device that has become an essential tool in almost every phase of speech research. In this device, a short speech utterance repeatedly modulates a variable frequency oscillator. The modulated signal is input to a bandpass filter. The average energy in the output of the bandpass filter at a given time and frequency is a crude measure of the time dependent Fourier transform. This energy is recorded by an ingenious electromechanical system on teledeltos paper. The result is called a spectrogram [22].

A spectrogram is a two dimensional representation of the time dependent spectrum in which the vertical dimension represents frequency and the horizontal dimension represents time. The spectrum magnitude is represented by the darkness of the marking on the paper. If the bandpass filter has a wide bandwidth (e.g. 300 Hz), the spectrogram displays good temporal resolution and poor frequency resolution. On the other hand, if the bandpass filter has a narrow bandwidth (e.g. 45 Hz), the spectrogram has a good frequency resolution and a poor time resolution. Examples are shown in Figure 3.6 and Figure 3.7.

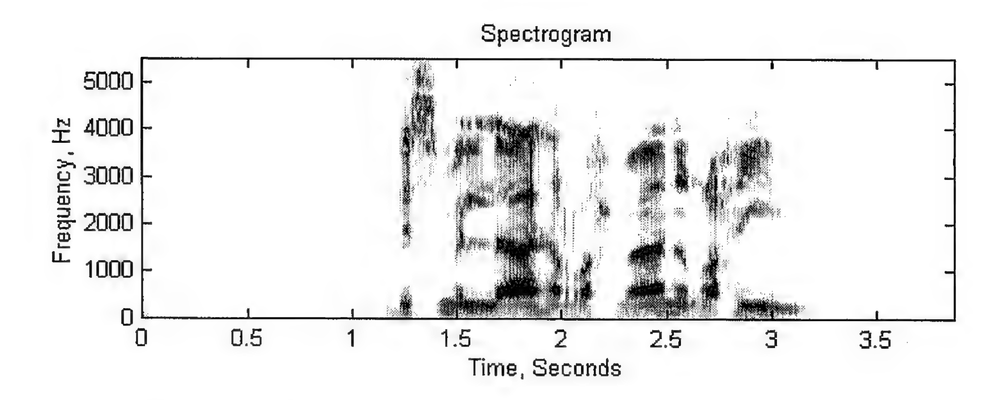



Figure 3.6 Wideband spectrogram (Window width; N=95)

Figure 3.6 shows a wideband spectrogram of the utterance " بسم الله الرحمن الرحيم ". This example illustrates a number of characteristic features of wideband time dependent spectra. The spectrogram clearly displays the variation of the formant frequencies with time. Another interesting feature of the wideband spectrogram is the vertical striations that appear in regions of voiced speech. These are due to the fact that the impulse response of the analyzing filter (i.e., the spectrum analysis window) duration is less than the pitch period. For unvoiced speech, which is not, of course, periodic, the vertical striations do not appear and the spectral pattern is much more ragged.

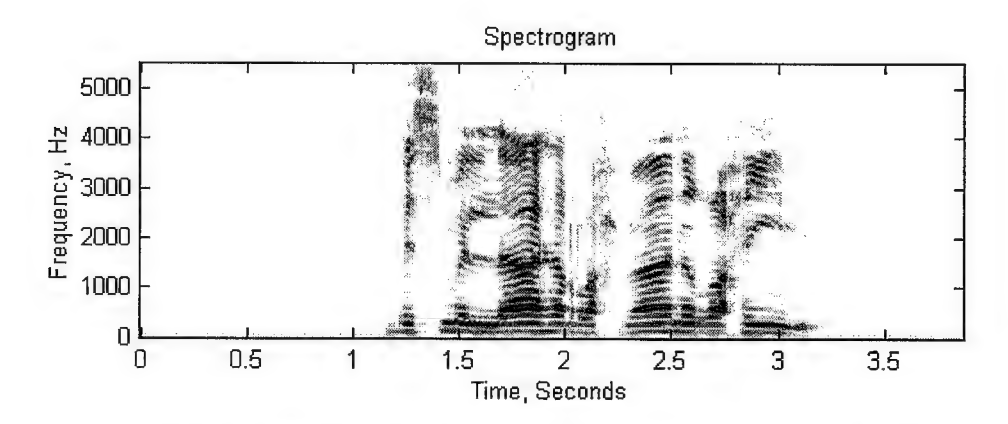



Figure 3.7 Narrowband spectrogram (Window width; N=256)

Figure 3.7 is a narrow band spectrogram of the same utterance. In this case, the bandwidth of the filter is such that individual harmonics are resolved in voiced regions. Unvoiced regions are distinguished by a lack of periodicity in the frequency dimension.

The wideband and the narrowband spectrograms display a great deal of information about the properties of a speech utterance. Indeed, when the apparatus

for displaying such ST Fourier representations first became available, it was hoped that such a display could provide a new "language" for communication with the deaf. In the years since this early work, many speech researchers have made measurements by hand on spectrograms to determine speech parameters such as formant frequencies and the pitch [22].

## 3.11 Formant Extraction Using Linear Prediction Spectra

The speech waveform can be modeled as the response of a resonator (the vocal tract) to a series of pulses (quasi-periodic glottal pulses during voiced sounds, or noise generated at a constriction during unvoiced sounds). The resonances of the vocal tract are called formants, and they are manifested in the spectral domain by maximum energy at the resonant frequencies.

The frequencies at which the formants occur are primarily dependent upon the shape of the vocal tract, which is determined by the positions of the articulators (tongue, lips, jaw, etc.). In continuous speech, the formant frequencies vary in time as the articulators change position.

### 3.11.1 Peaks vs. Poles

Clearly, an obvious method for extracting formants from LP would be to solve for the poles of the filter by setting the denominator in (3.23) to zero and solving for the roots of the resulting $p$th order real polynomial in z. Some or none of the roots would be real, and the rest would be complex conjugate pole pairs which might or might not be formants. Out of those pole pairs, one would have to select three on the basis of frequency location, sufficiently narrow bandwidth, and some kind of formant continuity criterion, to be the first three formants.

Another technique, requiring much less computation, would be to simply pick the first three peaks in the spectrum and call those the first three formants, making the assumption that a pole strong enough to show up as peak is necessarily a formant. Such a method works very well most of the time, but mistakes will occur during the following situations.

1) Often two poles show up as only one peak because they are close together in frequency.
2) Occasionally a pole due to frequency shaping will appear as a small peak, which would be incorrectly interpreted as a formant.

It was decided to use peak-picking rather than root extraction, and to develop an algorithm to solve cases 1) and 2) above. [2]

### 3.11.2 Algorithm steps

After speech classification voiced frames are used only. At each frame one begins with four vacant formant slots; $S_1$, $S_2$, $S_3$ and $S_4$, four estimates for the frequencies of the formants; $EST_1$, $EST_2$, $EST_3$, $EST_4$, and one, two, three, or four peaks. The task is to fill the slots with the peaks, based on the estimate frequencies, in such a way that spurious peaks and missing peaks can be recognized as such and dealt with. (No special attempt is made to fill the $F_4$ slot. It only exists to prevent $F_4$, when it exists, from competing with $F_3$ for the $F_3$ slot).

Steps of the peak mapping algorithm are explained below, and Figure 3.8 shows the flowchart of the peak mapping algorithm.

***Fetch Peaks***

1. Find the frequencies and amplitude of up to four peaks in the region from 150 to 3400 Hz for the segment spectrum obtained using LP with $r$=1.

***Fill Slots***

2. Fill each formant slot $S_i$, $i = 1$ to 4, with the best candidate peak $P_j$, where the peak $P_j$ closest in frequency to estimate $EST_i$ goes into slot $S_i$.

***Deal with Unassigned Peaks***

3. If there are no unassigned peaks $P_j$, go to 8.
4. Otherwise, try to fill empty slots with peaks not assigned in 2; If there is a peak $P_{j=k}$ unassigned, and an $S_{i=k}$ unfilled, fill the slot with the peak and go to step 8.
5. If there is a peak $P_{j=k}$ unassigned, but slot $S_{i=k}$ is already filled, check the amplitude of $P_k$: If amp ($P_k$) < ½ amp (peak already assigned to $S_k$) throw $P_k$ away and go to 8. Otherwise, go to 6.
6. If $P_k$ is still unassigned, but $S_{i=k+1}$ is unfilled, move the peak in $S_{i=k}$ to $S_{i=k+1}$, and put $P_k$ in $S_k$. Go to 8.
7. If $P_k$ is still unassigned, but $S_{i=k-1}$ is unfilled, move the peak in $S_{i=k}$ to $S_{i=k-1}$, and put $P_k$ in $S_k$. Go to Step 8. If 4, 5, and 6 all fail, throw $P_k$ away.

*Deal with Unfilled Slots*

8. If $S_1$, $S_2$ and $S_3$ are all filled, go to 13. ($F_4$ may or may not be filled).
9. Otherwise; Recompute the spectrum on a circle with radius less than one ($r$=0.98 ) to enhance the formants. Go to 1.
10. If the spectrum fails to yield a peak to fill the empty slot then step 1-9 are repeated again with $r = r - 0.004$. The radius is shrunk repeatedly in this manner until a peak is finally found or until $r = 0.88$; at which point it is assumed that no peak exists to fill the empty slot.
11. Finally, whether or not enhancement has succeeded, the amplitude of the peaks are reset to the amplitudes in the original spectrum.
12. If the empty slot was $S_3$, and failed to yield a peak, then the peak in $S_4$ is moved down to $S_3$, assuming that $F_3$ was mistakenly called $F_4$.

*Record Answers*

13. Accept formant slot contents as answers, and if a slot is empty, keep the original formant estimate for that formant [2].

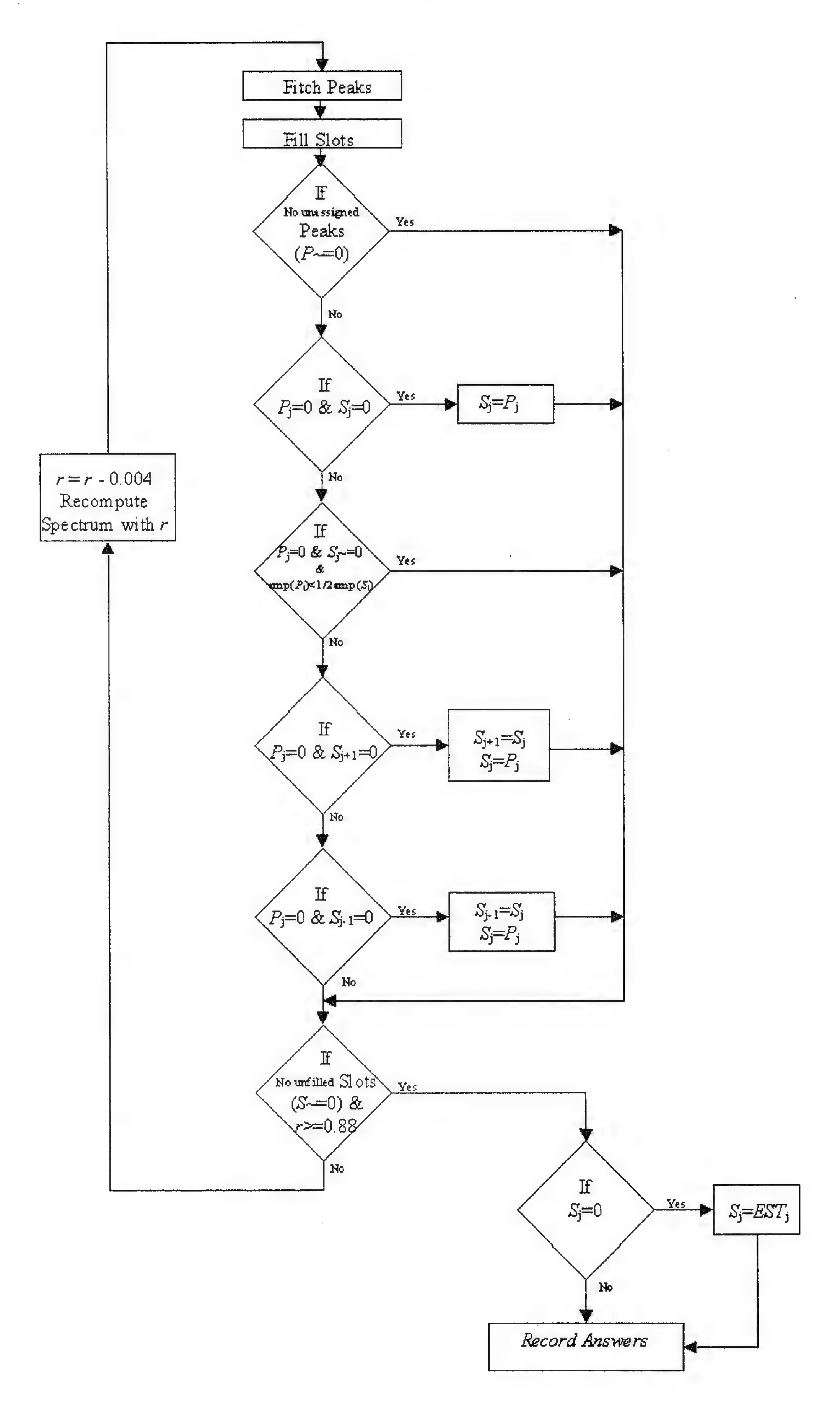



Figure 3.8 Peak mapping algorithm

# CHAPTER IV
# GRAPHICAL USER INTERFACE (GUI)

## 4.1 Introduction

*Guide* extends MATLAB's support for rapid coding for building GUIs. Guide is a set of MATLAB tools designed to make building GUIs easier and faster. Just as writing math in MATLAB is much like writing it on paper, building a GUI with Guide is much like drawing one on paper. As a result, one can lay out a complex graphical tool in minutes. Once buttons and plots are in place, the Guide Callback Editor sets up the MATLAB code that gets executed when a particular button is pressed.

## 4.2 Design Principles

Designing means everything can be done before writing the code that results in a final working GUI. Most important principles in design are Simplicity, Consistency, and Familiarity.

- **Simplicity:**

A simple GUI has a clean look and a sense of unity. It's very easy to add functionality to the GUI, but if that functionality really doesn't belong, it should be taken out. Avoid screen clutter, and only present users with choices that advance them toward the completion of the task. Once we let ourself remove a piece of the GUI that doesn't absolutely need to be there, we may find that we can eliminate a lot of supporting machinery that no longer has any purpose.

- **Consistency:**

The further users are from their base of experience, the more likely they are to feel disoriented. Anything we can do to keep the user from feeling confused is extraordinarily important.

- **Familiarity:**

If the GUI is in some sense familiar to its users, then they can generally learn how to use it more quickly. This is the value of basing the GUI on a good metaphor. The

users might not know how to do a given task, but the metaphor helps them make a good guess. Familiarity draws people into the GUI and makes them feel comfortable.

## 4.3 The Dynamic Interface

Actions on the GUI should be: Immediate, Continuous, and Reversible. With respect to immediacy and continuity, calculation time is the most important consideration. Calculation and display the result should be made instantly. But if calculation time is significant, we may be better off using a button to invoke the action. Finally, reversibility is most often embodied by the Undo menu. Undoing is sometimes difficult to implement, but it's always appreciated by users. A well-built undo capability encourages experimentation and a sense of comfort in working with the GUI.

## 4.4 Design Process

It's helpful to think about the GUI creation process as breaking into a design phase and an implementation phase. The fact presented below is complete the design of the GUI before the beginning of implementing the GUI.

*Coding must be after designing.* If we start working on the implementation too soon, we tend to converge much slower on a final design.

Of course the design may change once we start coding, and there may well be design decisions that we can't properly make until we have written some code. Still, take the design part just as far as we can on paper, because we are guaranteed to save ourselves a lot of time.

*Start with the ideal.* Don't make compromises too soon. The best final designs grow form an idealized initial vision.

The ideal design is something that exists in our head after we have spent a long time thinking about the task and the user. This design might be expensive in terms of development effort or computing power required. But for whatever reason, it is the best way to approach the task. Think of it as a three dimensional shape that must now be projected down into the two dimensions of the final implementation. If we start out with the merely achievable, the final result will be half-hearted. If we start with the ideal, it is often surprising what we can make the technology do for us.

*Test the GUI on Paper.* Drawing and testing the GUI on paper keeps us and any testers from over focusing on the detailed aspects of the GUI.

Early on in the design process, there is no point in playing with font sizes and button colors. What matters is the overall behavior and appearance of the GUI. Paper prototyping is a good exercise for keeping things in perspective. The idea is to build the entire interface out of paper and try it out. Since there is no code involved, we play the role of the computer while a cooperative user sitting across from us performs a task. It is an excellent way to see if the GUI does what we (and the user) want it to do.

Paper prototyping tests the thoroughness of the design and can resolve disagreements about tough design questions that might otherwise be intractable.

By the time we are ready to start writing code, we should have a complete diagram or set of diagrams of the GUI (the layout) and an exact description of what functionality is associated with each part of the GUI (the callbacks). Actually, the term "write the code" is somewhat misleading now that *Guide* is available to make the job easier.

## 4.5 Handle Graphics

All objects appears in a MATLAB figure is an example of Handle Graphics, because every object on the screen has a unique identifier, called a handle, that allows the designer to go back and modify the object at any time. Figure 4.1 is a diagram of the Handle Graphics object hierarchy, that shows all the categories of object types in MATLAB.

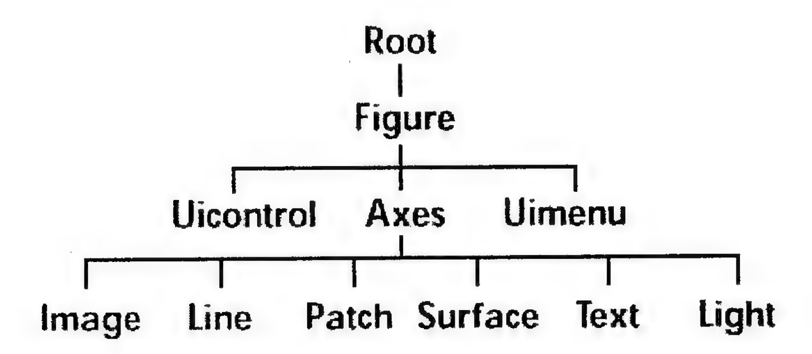



Figure 4.1 Handle graphics object hierocracy

## 4.6 Speech Analysis GUI

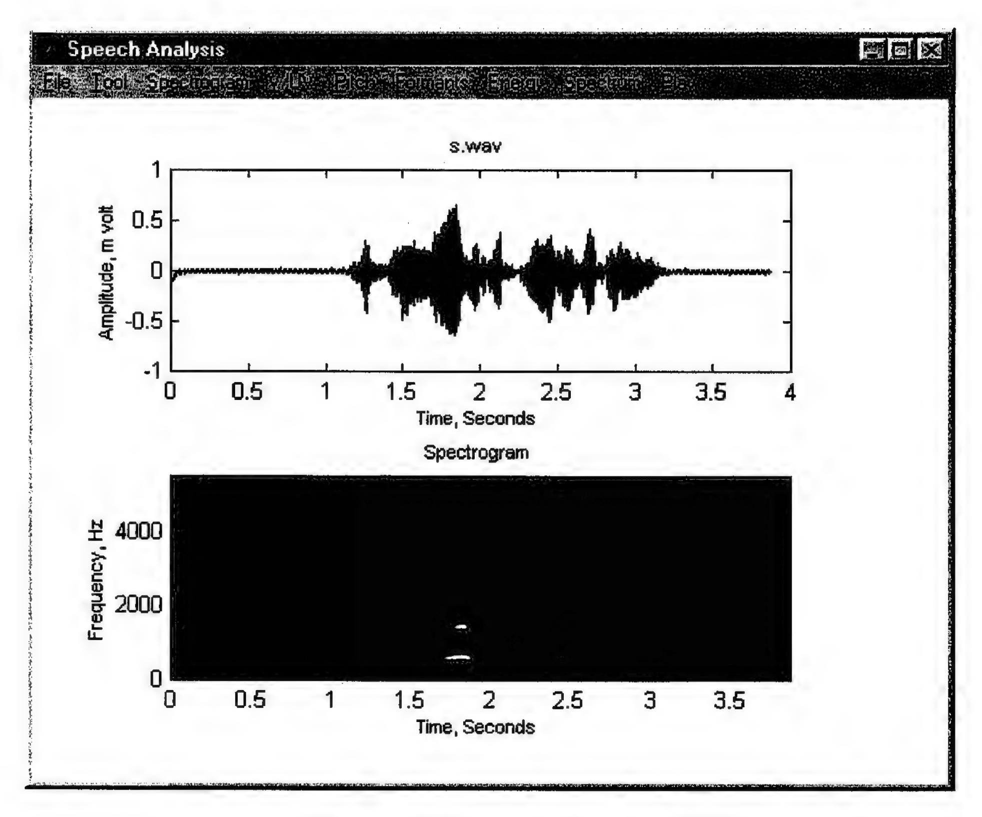
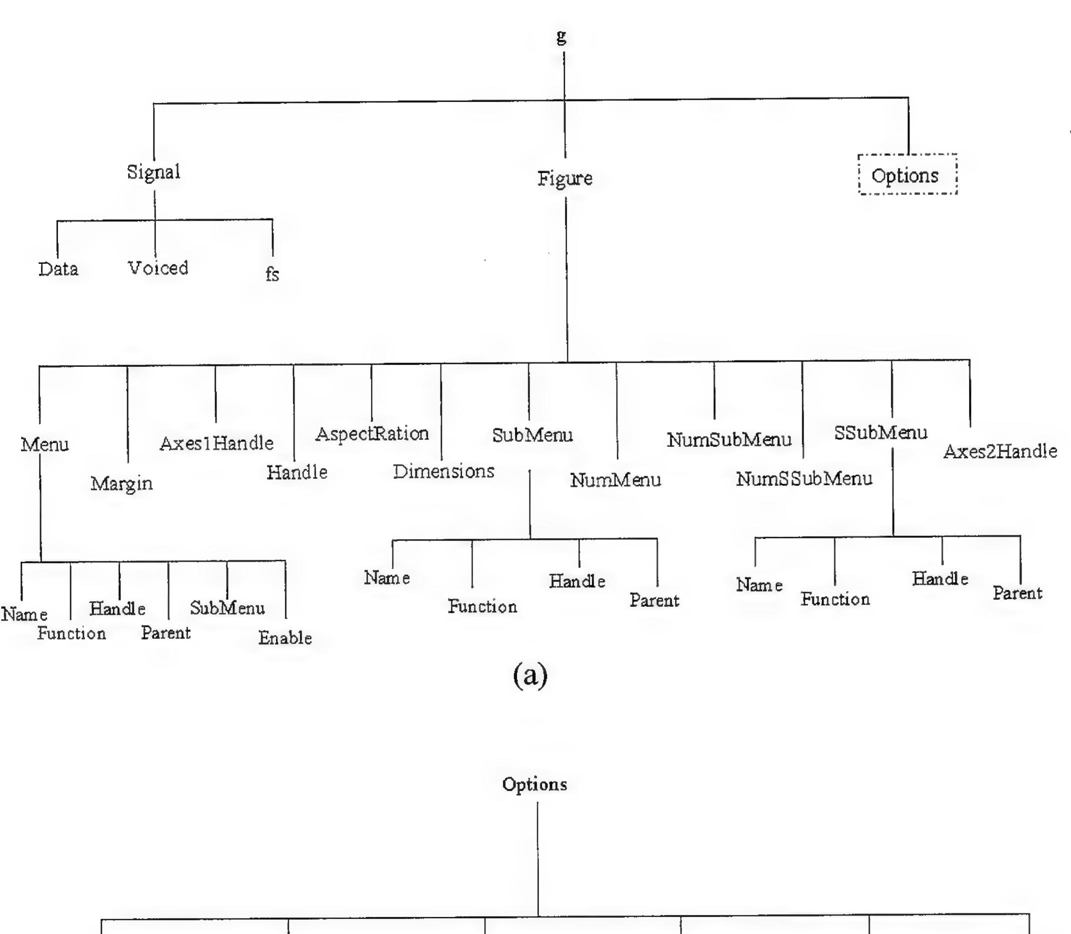



Figure 4.2 Speech Analysis GUI

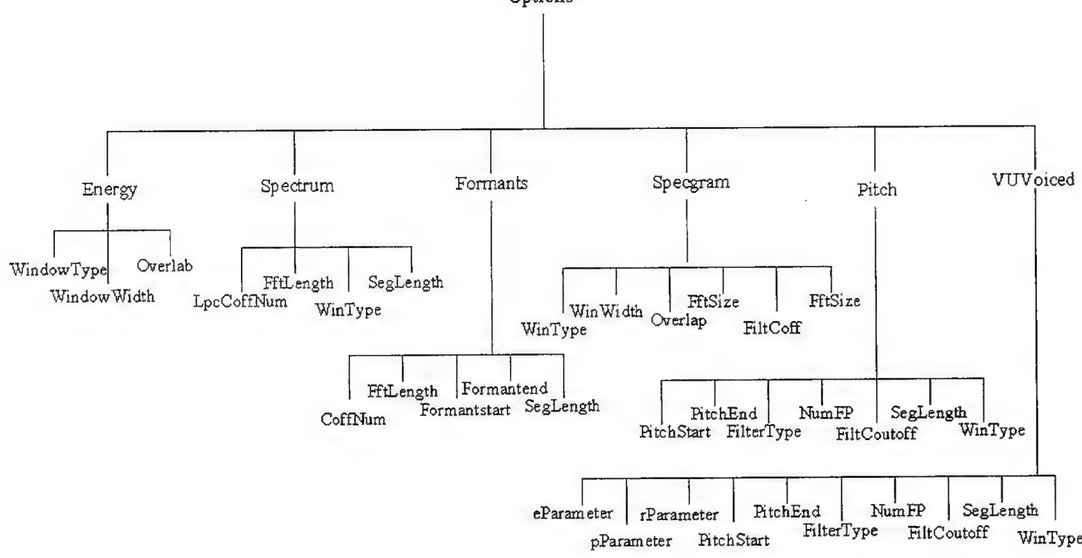
Figure 4.2 shows Speech Analysis GUI. It consists of a Figure object which is the parent of all the menus and axes objects. Under some menus there are submenus and then sub-submenus where the parent of submenu is menu. Each object has a certain handle and properties where through its handle, we can get and set any property belong to this object. In an other word we can control this object and put it in the way which can help the user.

## 4.7 Basic Idea

By using *struct* keyword we can create a structure array with specified fields and values. I define my structure as global in scope. This is because, ordinarily, each MATLAB function, defined by an m-file has its own local variables, which are separate from those of other functions and from those of the base workspace. However, if several functions, and the base workspace, all declare a particular name as global, they all share a single copy of that variable. Any assignment to that variable, in any function is available to the functions declaring it global. I call my structure g and all variables are involved in g.




(a)




(b)

Figure 4.3 The g structure tree

Global structure g has three basic variables which are Signal, Figure and Options. Signal has all variables belong to the speech we want to analyze. Figure contains all objects appear on the screen, each object has their properties as shown in Figure 3.4 (a). Options include all analysis methods like spectrogram, Energy, Formants, Pitch, Voiced/Unvoiced and Spectrum. Each of these analysis has it's own variables like in Energy, it has Window Type, Window Width and overlap

global variable as shown in Figure 4.3 (b). Some of these variables has a default value, Variables setting with default values is performed in the beginning of program.

### 4.8 GUI code

In general any code should be clear, understandable, strong and changeable without problem arises. Using the *Guide* control panel to construct GUI is useful. The disadvantage of Guide is the layout of graphic objects is too long, mixed, weak and not suitable for tough GUI.

Speech Analysis GUI code is short, clear, strong, understandable. It has the ability to add a new analysis method without big effort or many changes. Also the addition of a new menu, submenu and sub-submenu can be done smoothly.

# CHAPTER V
# SPEECH ANALYSIS SOFTWARE

## 5.1 Introduction

Speech analysis software consists of the main window which have nine menus and two axes. The input is a speech signal using pieces of Personal Computer (PC) hardware such as a microphone and a sound card. The recorded speech is processed then the output is shown on the two axes and heard through the speaker. Figure 5.1 shows speech analysis software with outputs on axes.

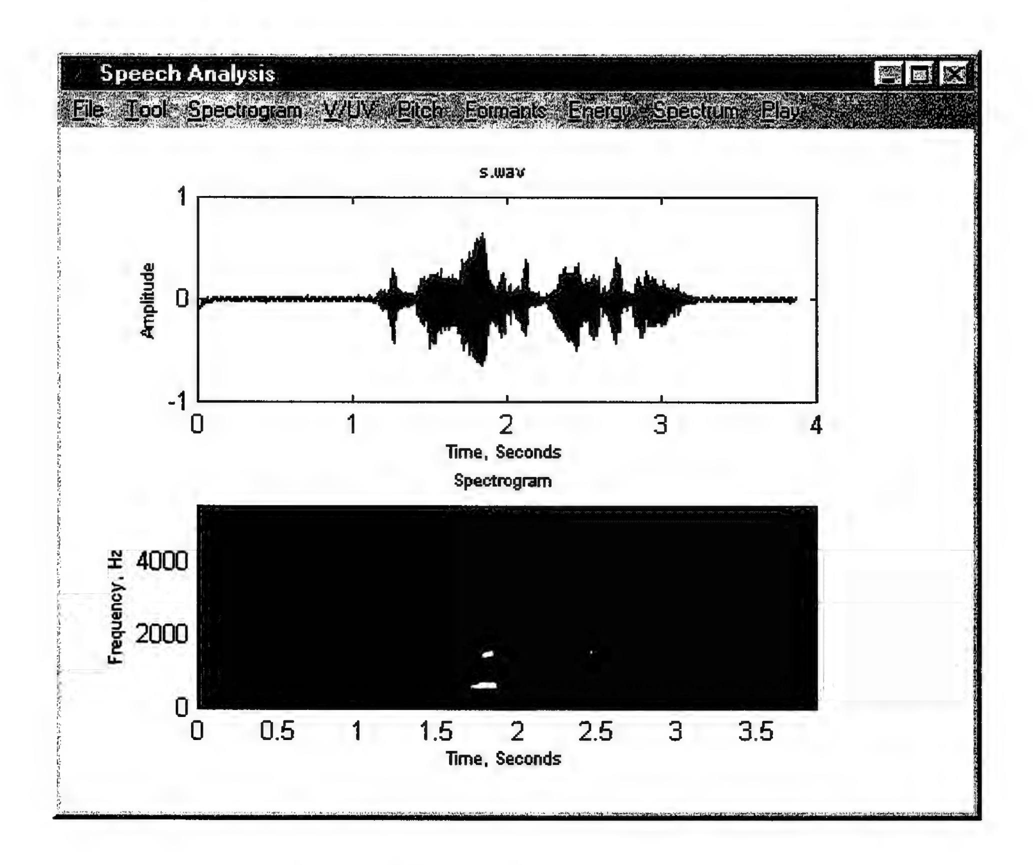



Figure 5.1 Speech Analysis software

## 5.2 File menu

With this menu the user start analysis, where in the beginning all menus are off except file menu. File menu consists of three submenus; *Open* to select a recorded sound wave as shown in Figure 5.2, *New* to record a new sound wave, and finally *Exit* to close speech analysis software.

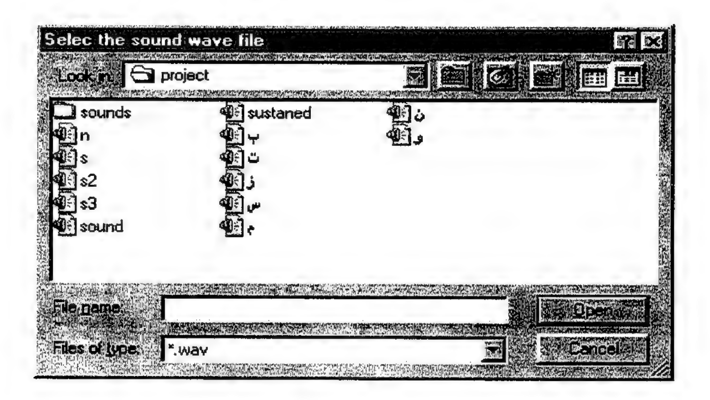



Figure 5.2 File menu

After file selection the waveform of the selected file appears on the top axes and all menus are enabled on.

## 5.3 Spectrogram menu

Spectrogram menu has two submenus, *Spectrogram* and *Color.* Using *Spectrogram* submenu the spectrogram of the sound wave is shown on the bottom axes. *Color* gives you the ability to change the colormap of the spectrogram. Under this submenu there are five sub-submenus, gray, hot, hsv, cool, and jet as shown in Figure 5.3.

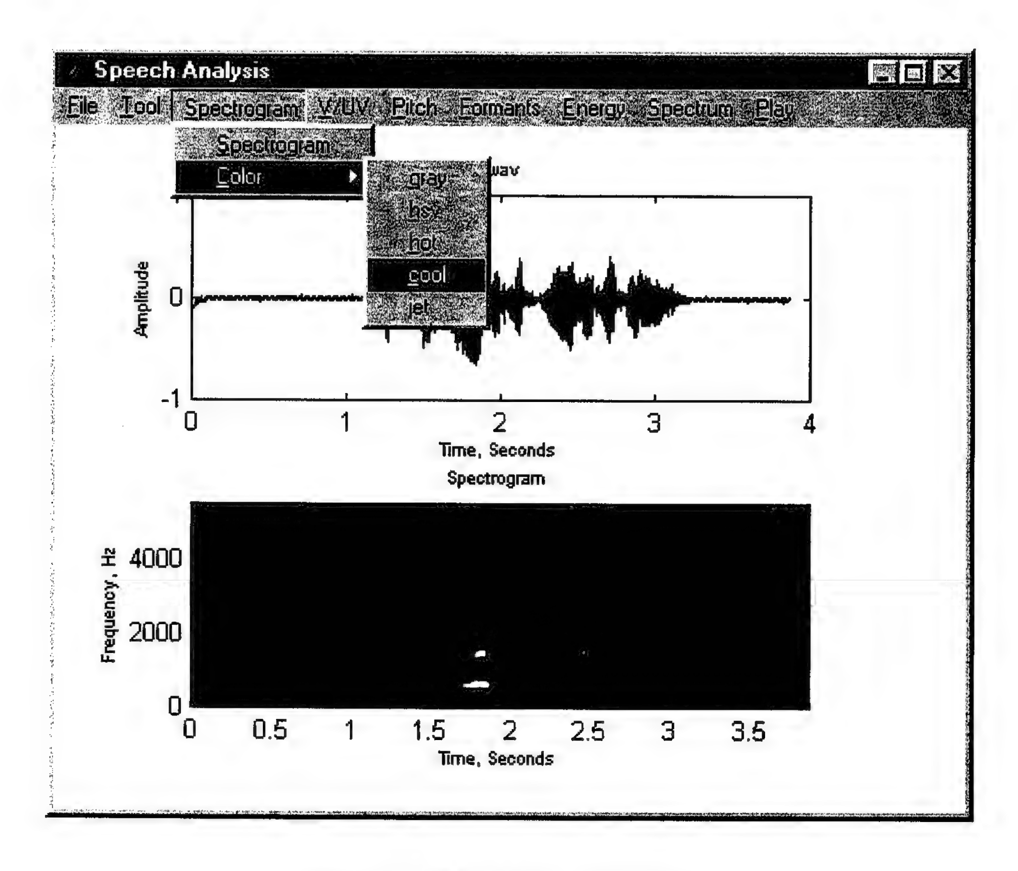



Figure 5.3 Spectrogram menu

## 5.4 V/UV menu

V/UV menu makes voiced and unvoiced classification. On the top axes voiced segments are represented by red color where unvoiced segments are represented by black as shown in Figure 5.4.

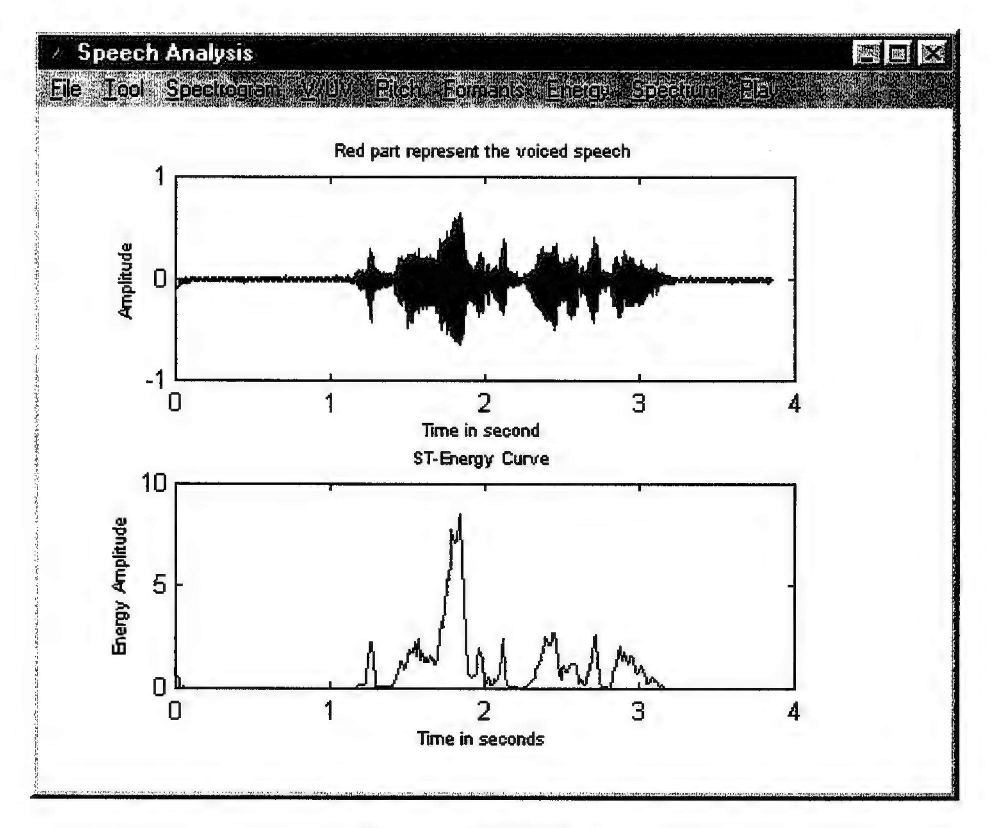



Figure 5.4 V/UV menu

## 5.5 Pitch menu

Pitch menu has three submenus, *Averaged Pitch*, *Single Pitch*, and *Pitch Curve*. *Averaged Pitch* gives the user the ability to select any part of the sound wave, *Averaged Pitch* divide this part to ST segments then computes the pitch for each of them and then shows the user the average of this computation. *Single pitch* submenu computes the pitch for a single short segment that is selected by the user. The autocorrelation function appears on the bottom axes as shown in Figure 5.5. *Pitch Curve* submenu computes the pitch over the whole sound wave, then the pitch curve appears on the bottom axes as shown in Figure 5.6.

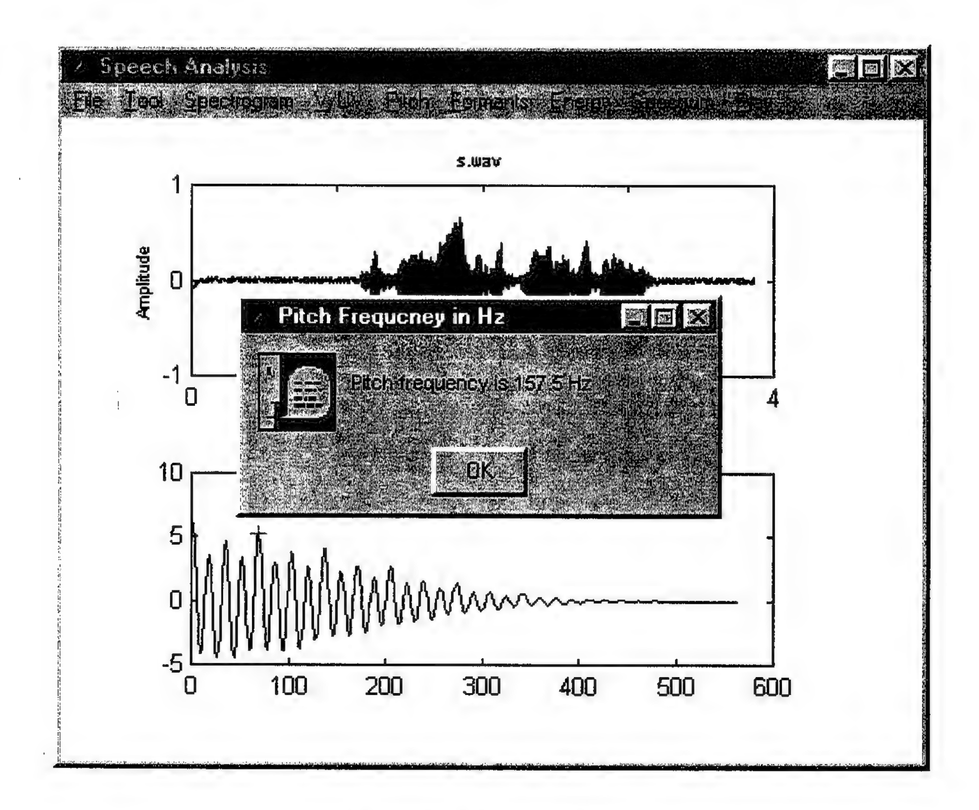



Figure 5.5 Pitch menu

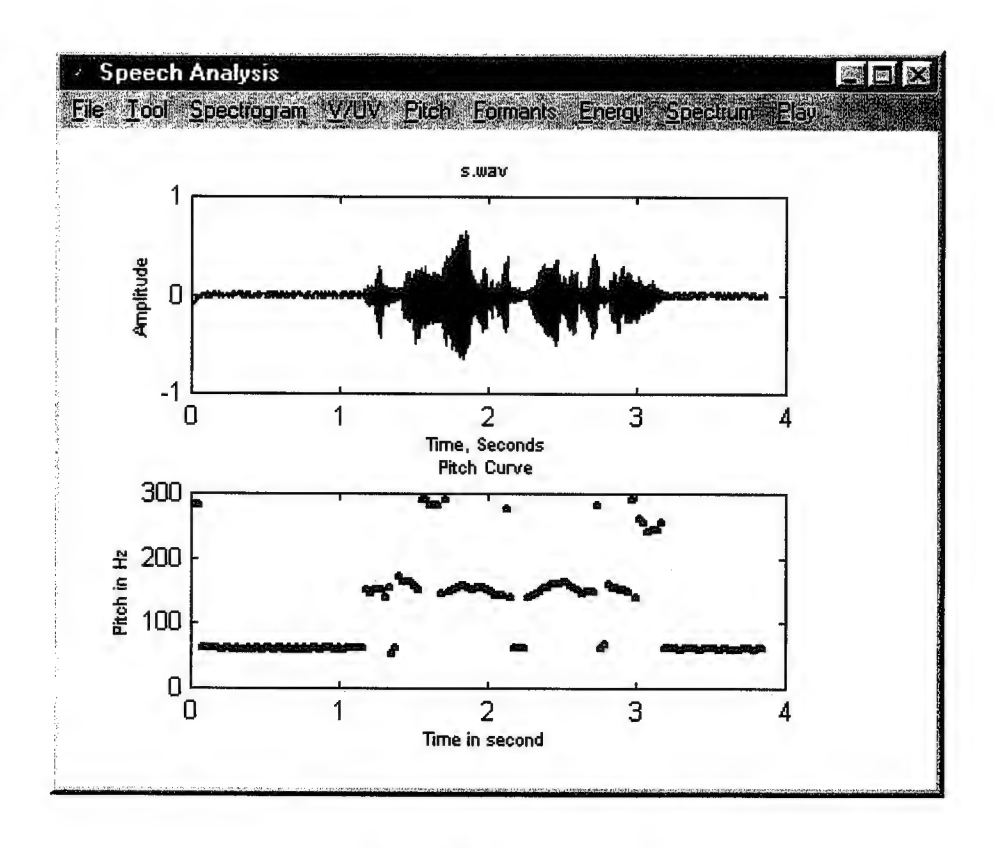



Figure 5.6 Pitch curve

## 5.6 Formants menu

Formants menu has two submenus, *Formants* and *Formants History*. *Formants* submenu computes the first three formants of any ST segment in the sound waveform as shown in Figure 5.7. *Formants History* computes the formant for the whole sound wave then plots all formants on the spectrogram as shown in Figure 5.8.

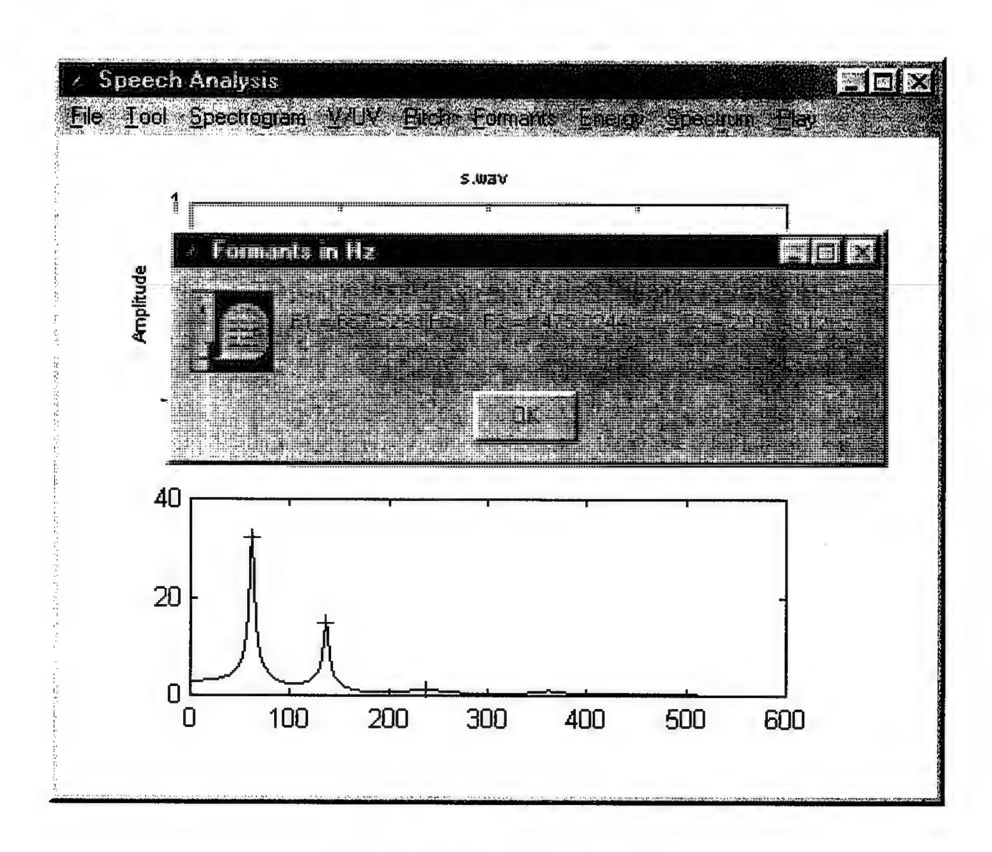



Figure 5.7 Formants menu

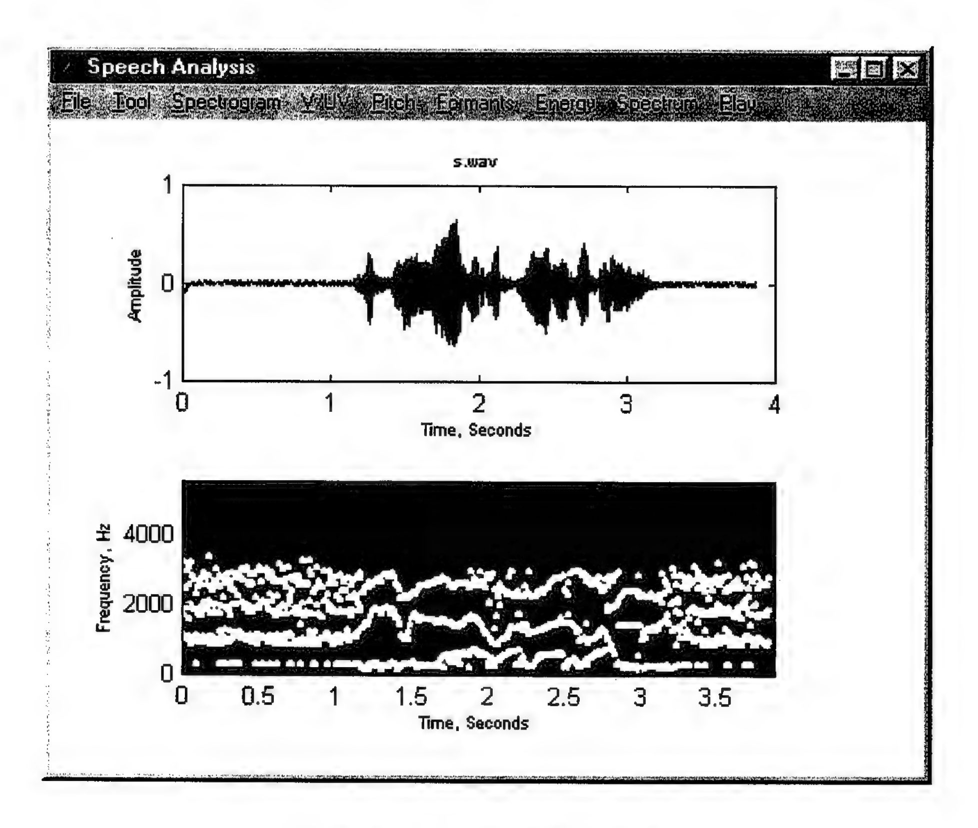



Figure 5.8 Formants History

## 5.7 Energy menu

Energy menu computes ST-Energy of the whole signal, and shows the user the energy profile on the bottom axes as shown in Figure 5.9.

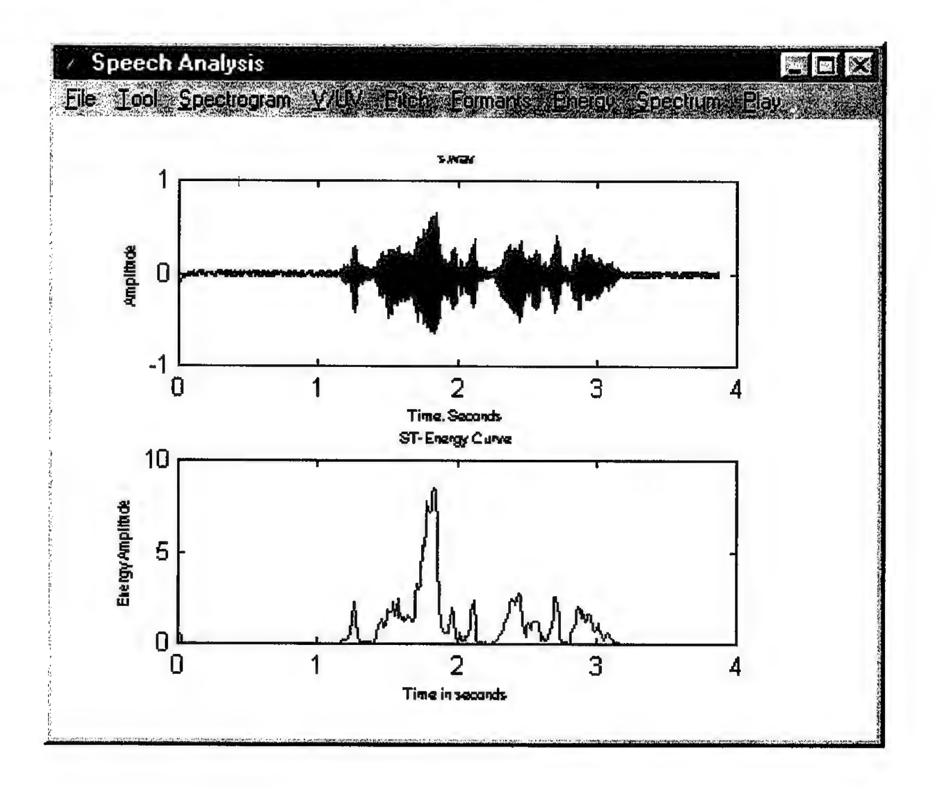



Figure 5.9 Energy menu

## 5.8 Spectrum menu

Spectrum menu computes the ST Fourier transform then computes spectrum smoothing using LP for any selected ST segment.

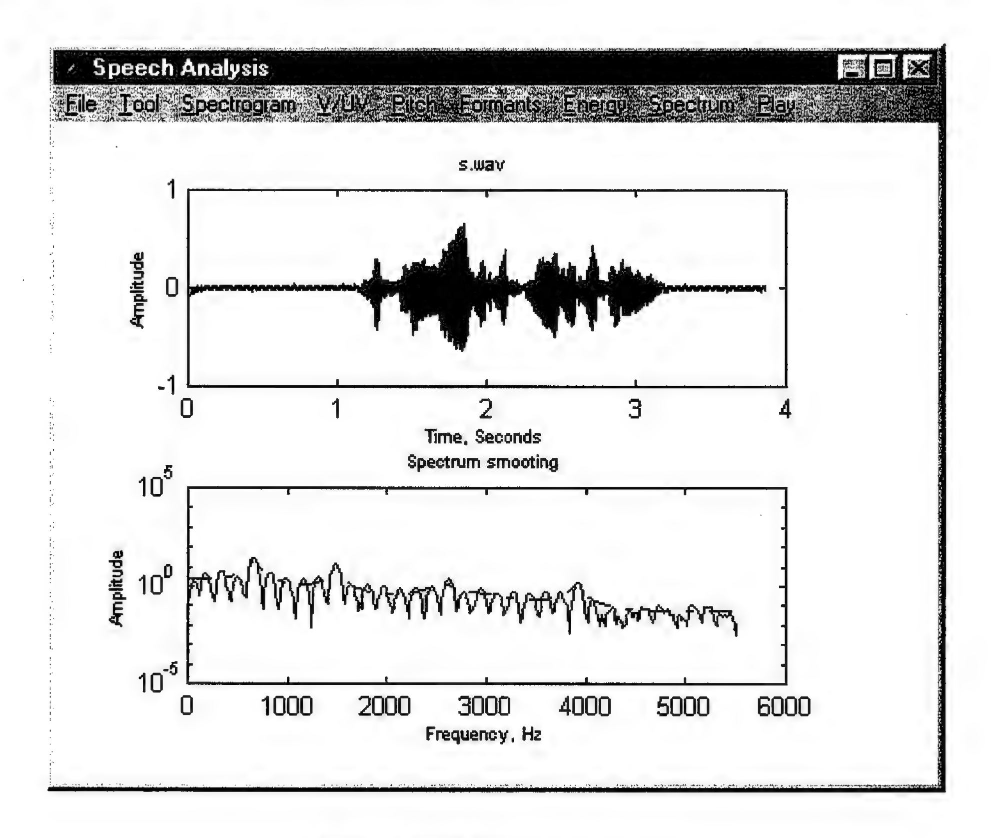



Figure 5.10 Spectrum menu

## 5.9 Tool menu

Tool menu has two submenus *Reset Defaults* and *Options*. *Reset Defaults* to reset all analysis options to default values. *Option* has five sub-submenus as shown in Figure 5.11.

The five sub-submenus are explained below:

*Spectrogram*, through which the user can set the following spectrogram options:

1. Window length in points
2. Overlap in points
3. FFT length in points
4. Window type

Figure 5.12 shows Spectrogram Options.

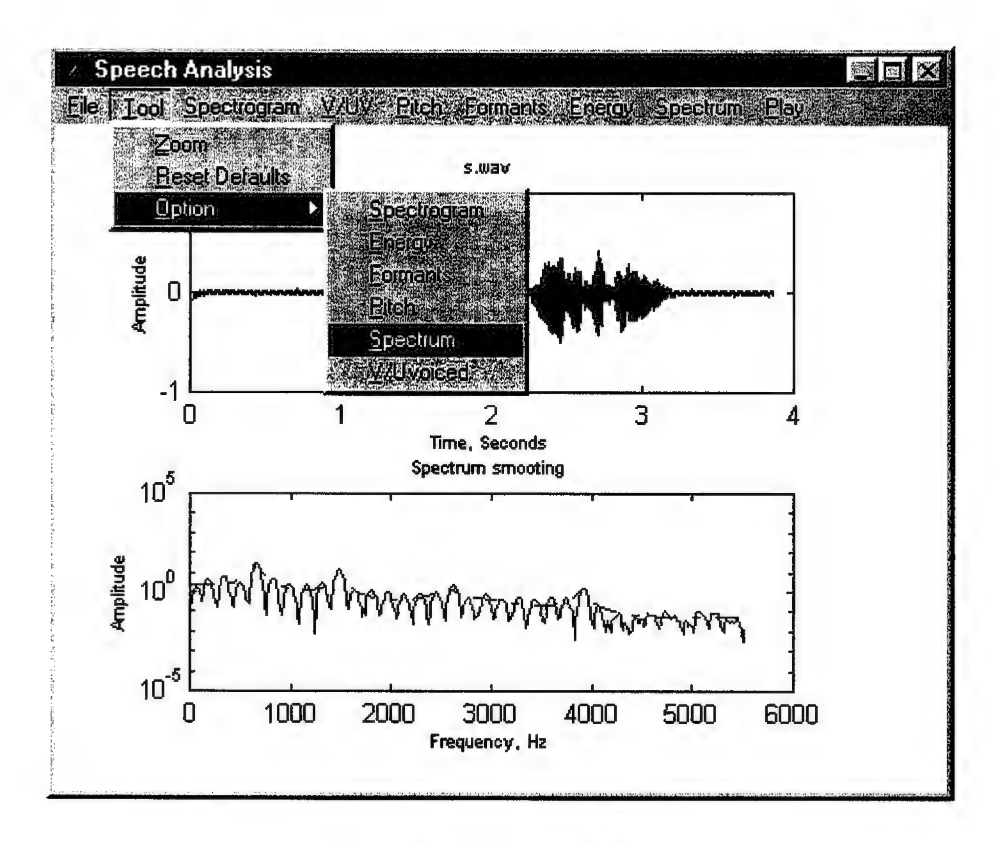



Figure 5.11 Tool menu

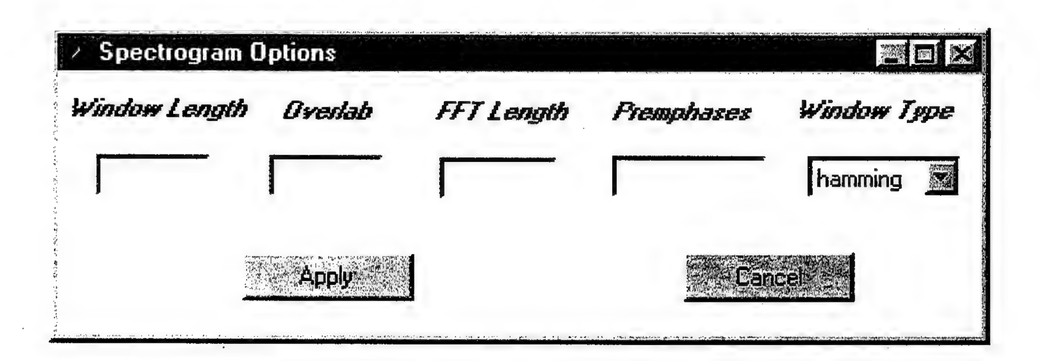



Figure 5.12 Spectrogram Options

*Energy*, the user can set the following energy options:

1. Window length in points
2. Overlap in points
3. Window type

Figure 5.13 shows Energy Options.

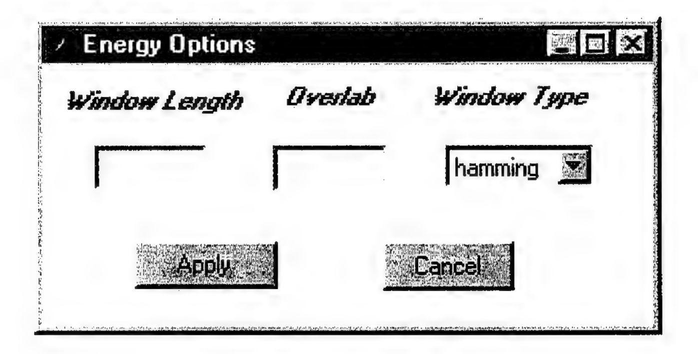



Figure 5.13 Energy Options

*Formants,* the user can set the following formants options:

1. Number of coefficients
2. FFT length in points
3. Formants start in Hz
4. Formants end in Hz
5. Segment length in ms

Figure 5.14 shows Formants Options.

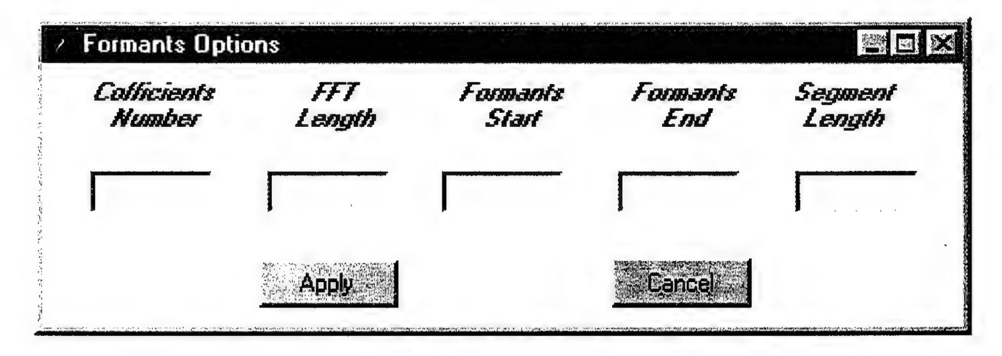



Figure 5.14 Formants Options

*Pitch*, the user can set the following pitch options:

1. Pitch start in Hz
2. Pitch end in Hz
3. Segment length in ms
4. Number of filter poles
5. Filter cutoff frequency
6. Window overlap
7. Filter type
8. Window type

Figure 5.15 shows Pitch Options.



Figure 5.15 Pitch Options

*Spectrum*, the user can set the following spectrum options:

1. Number of LP coefficients
2. FFT length in points
3. Segment length in ms
4. Window type

Figure 5.16 shows Spectrum Options.

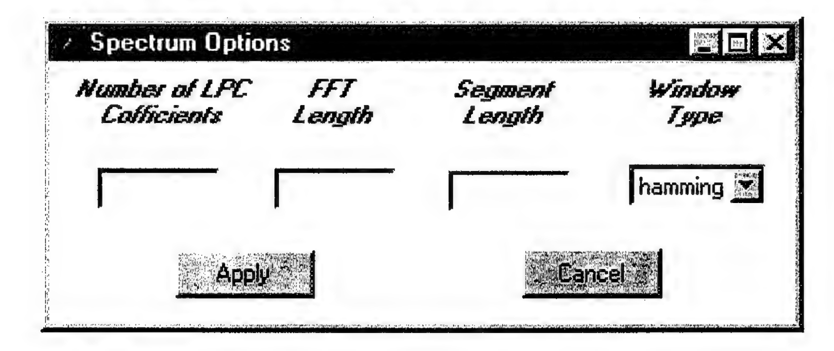



Figure 5.16 Spectrum Options

*V/Unvoiced*, the user can set the following options:

1. e parameter
2. p parameter
3. r parameter
4. Pitch start in Hz
5. Pitch end in Hz
6. Segment length in ms
7. Number of filter poles
8. Filter cutoff frequency
9. Window overlap
10. Filter type
11. Window type

Figure 5.17 shows V/UVoiced Options.

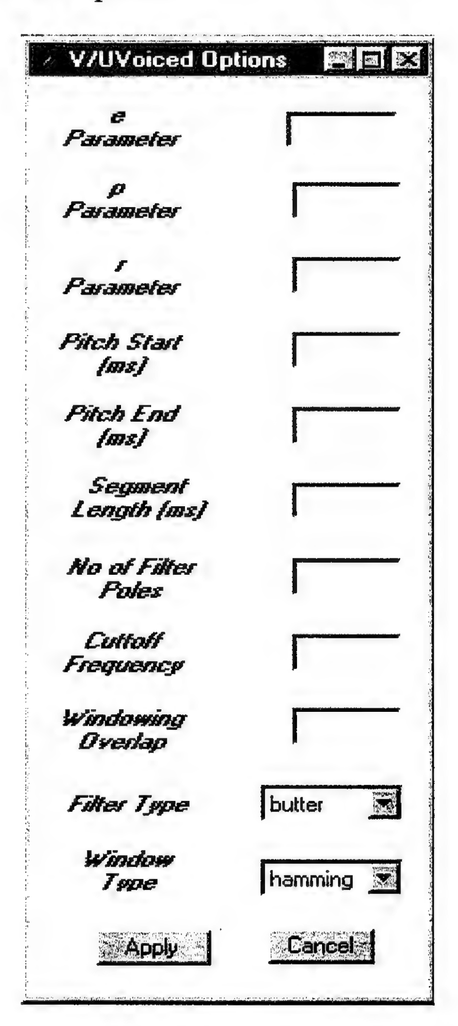



Figure 5.17 V/UVoiced Options

In all the above options the user can implement the setting by pressing Apply button.

## 5.10 Play menu

Play menu consists of three submenus, *All Data*, *Voiced Segments* and *Data Segment* as shown in Figure 5.18. *All Data* submenu sounds the whole waveform, *Voiced Segments* sounds voiced segments only, and *Data Segment* submenu gives the user the ability to select a part from the waveform then sounds the selected part.

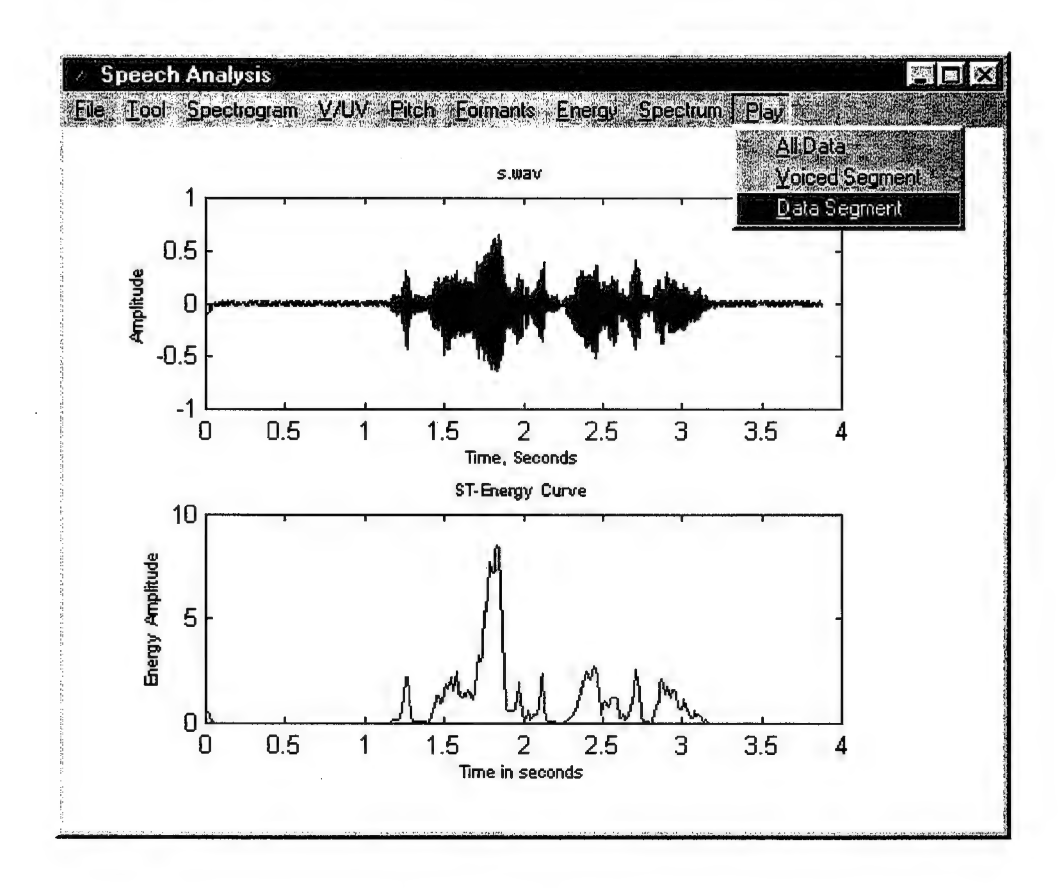



Figure 5.18 Play menu

## 5.11 Some application of Speech Analysis Software

Speech and acoustic analysis have increased use in the evaluation of patients with various voice disorders.

***V/UV application:***

V/UV classification can be used by speech therapists to evaluate patient voice. Some people have voice disorder, for instant they produce voiced sounds as unvoiced or vise versa. The patient can record a phrase, then the speech pathologist can determine the sound which is devoiced or made voiced through seeing the colors that represent voiced and unvoiced.

A patient of hyper nasality for example had a problem in producing the voiced counterpart of certain sounds such as t, k. By watching this classification speech Language Pathologist (SLP) can tell the patient where he should see red color for voiced sounds. SLP can record his own speech to compare and see where he should see red color for voiced sounds.

***Pitch application:***

Some patients have voice disorders related to pitch. Pitch means how the person speech sounds. If the patient is an adult and still not develop an adult pitch, his speech would sound as a child's speech, "The perceptual effect is a boy's voice in a man's body".

SLP can use pitch to determine habitual pitch; the pitch that patient actually uses. Then it can be used to reach the optimal pitch; the pitch that is suitable for the patient age. The patient tries to change his pitch and watch the change on the PC.

***Energy application:***

Some patients have low voices, so if they record they will have low energy curve. After giving these patients certain exercises to increase their sound energy, they can use energy curve to feel the difference. However SLP can use energy curve as a training tool for patient to increase his energy.

# CHAPTER VI

# CONCLUSION & FUTURE WORK

In this thesis speech production is explained, from this we see some of what Allah subhanah give us in this field. Speech processing and analysis; algorithms and mathematical background is also explained.

A complete speech processing and analysis software is generated. This software based on MATLAB, and has the following analysis: Energy, Spectrogram, Pitch, Formants, Voiced/Unvoiced classification, and Spectrum. Each of these analysis has it's own options and parameters to change and control the analyzed operation.

Future work suggestions are:

a. Converting the whole speech analysis software to C++ language to make a stand alone software.
b. Adding others methods to extract the same feature. Like in pitch detection, we used Autocorrelation method, In the future we can add Cepstrum and AMDF methods for pitch extracting.
c. Classify the speech to voiced, unvoiced and silence segments.
d. Adding the continuity of the pitch curve to the Pitch analysis.
e. Adding the continuity of the formant history to the Formant analysis.
f. Adding short time Cepstrogram analysis
g. Using Wavelet transform as an analysis method.
h. Combining a microphone and a recorder with the speech analysis software.

# REFERENCES

1) Adel M. Abu-Shaar, Private Communication.
2) S. S. McCAnndless, “An Algorithm for Automatic Formant Extraction Using Linear Prediction Spectra,” *IEEE Trans. On Acoust., Speech, and Signal Process.*, vol. ASSP-22, pp. 135-141, Apr. 1974.(1)
3) D. A. Krubsack and R. J. Niederjohn, “An Autocorrelation Pitch Detector and Voicing Decision with Confidence Measures Developed for Noise-Corrupted Speech,” *IEEE Trans. On Signal process.*, vol. 39, no. 2, Fep. 1991.(4)
4) L. R. Rabiner, M. J. Cheng, A. E. Rosenberg and C. A. Gonegal, “A Comparative Performance Study of Several Pitch Detection Algorithms,” *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. ASSP-24, pp. 399-417, Oct. 1976.
5) M. M. Sondhi, “New methods of pitch extraction,” *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 262-266, June 1968.
6) B. Gold and L. Rabiner, “Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain,” *J. Acoust. Soc. Am.*, vol. 46, pp. 442-448, Aug. 1969.
7) M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg, and H. Manley, “ Average magnitude difference function pitch extractor,” *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-22, pp. 353-362, Oct. 1974
8) M. Noll, “Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate,” in *Proc. Symp. Comput. Processing Commun.*, Apr. 1969, pp. 779-797
9) M. Lahat, R. J. Niederjohn, and D. A. Krubsack, “A spectral autocorrelation method for measurement of the fundamental frequency of noise-corrupted speech,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 741-750, June 1987.
10) M. Noll, “Cepstrum pitch determination.” *J. Acoust. Soc. Amer.*, vol. 47, pp. 293-309, Feb. 1967.
11) J. D. Markel, “The SIFT algorithm for fundamental frequency estimation,” *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 367-377,.Dec. 1972.

12) E. N. Pinson, "Pitch-Synchronous Time-Domain Estimation of Formant Frequencies and Bandwidths," *J. Acoust. Soc. Am.*, vol. 35, pp. 1264-1273, Aug. 1963.

13) R. W. Schefer, L. R. Rabiner, "System for Automatic Formant Analysis of Voiced Speech," J. Acoust. Soc. Am., vol. 47, pp. 634-648, Feb. 1970.

14) F. Itakura, S. Saito, "A Stiatistical Method for Estimation of Speech Spectral Density and Formant Frequencies," *Electron. And Commun.*, vol. 53-A, pp. 36-43, 1970.

15) L. J. Siegel, "A procedure for using pattern classification techniques to obtain a voiced/unvoiced classifier," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-27, pp. 83-89, Feb. 1979.

16) H. Kobatake, "Optimization of voiced/unvoiced decisions in nonstationary noise environments," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, pp. 9-18, Jan. 1987.

17) B. S. Atal and L. R. Rabinar, "A parrern recognition approach to voiced/unvoiced/silence classification with applications to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 201-212, June 1976.

18) C. K. Un and H. H. Lee, "Voced/unvoiced/silence discrimination of speech by delta modulation," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-28, pp. 398-407. Aug. 1980.

19) L. J. Siegel, A. C. Bessey, "Voiced/unvoiced/mixed excitation classification of speech," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-30, pp. 451-460, June 1982.

20) B. S. Atal, "Effectiveness of Linear Prediction Characteristics of the Speech Wave for Automatic Speaker Identification and Verification," *J. Acoust. Soc. Am.*, vol. 55, pp. 1304-1312, June 1974.

21) A. A. Almusleh, Speech Recognition. Master Thesis, KAAU, 1999.

22) L. R. Rabiner and R. W. Schafer, Digital Processing of Speech Signals. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1978.

23) J. L. Flanagan, "Automatic Extraction of Formant Frequencies from Continuous Speech," *J. Acoust. Soc. Am.*, vol. 28, pp. 110-118, Jan. 1956.

24) S. M. Kay, *Modern Spectral Estimation*. Prentice-Hall, Englewood Cliffs, 1988.

25) J. R. Deller, J. G. Proakis and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. Macmillan Publishing Company, New York, 1993.

26) R. W. Schafer and L. R. Rabiner, "System for Automatic Formant Analysis of Voiced Speech," *J. Acoust. Soc. Am.,* vol. 47, pp 634-648, Fep. 1970.

27) J. Ross, L Shaffer, R. Freudberg and H. J. Manley, "Average Magnitude Difference Function Pitch Extractor," *IEEE Trans. Acoust., Speech, and* L. R. Rabiner, "On the Use of Autocorrelation Analysis for Pitch Detection," *IEEE Trans. On Acoust., Speech, and Signal Process*., vol. ASSP-25, no. 1, Feb. 1977.

28) A. M. Noll "Cepstrum Pitch Determination," *J. Acoust. Soc. Am.,* vol. 41, pp. 293-309, Feb. 1967.

29) B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *J. Acoust. Soc. Am.,* vol. 50, pp. 637-655, Aug. 1971.

٣٠) نور الدين، عصام (١٩٩٢) علم وظائف الأصوات اللغوية الفونيتيكا، بيروت: دار الفكر اللبناني.

# APPINDEX A

## Thesis Matlab Programs

==================== *Basic Definitions*======================

```
function Def_menuNew
% This function contains a structure array which has a full
information
% about applied functions and operations on the speech signal and
Objects
% of Graphical user interface.

global g;


g=struct('signal',[],'Figure',[],'Options',[],'Defaults',[]);

%g.sinal
g.signal=struct('Data',[],'voiced',[],'fs',[]);

%g.Figure
g.Figure =
struct('Menu',[],'SubMenu',[],'SSubMenu',[],'Handle',[],'AspectRatio
',[],'Dimensions',[],'Margin',[],'NumMenu',[],'NumSubMenu',[],'NumSS
ubMenu',[],'Axes1Handle',[],'Axes2Handle',[]);

g.Figure.Menu=struct('Name',[],'Function',[],'Handle',[],'Parent',[]
,'SubMenu',[],'Enable',[]);

%g.SubMenu
g.Figure.SubMenu=struct('Name',[],'Function',[],'Handle',[],'Parent'
,[]);

%g.SSubMenu
g.Figure.SSubMenu=struct('Name',[],'Function',[],'Handle',[],'Parent
',[]);

%==================================================================

%g.Options
g.Options =
struct('Specgram',[],'Energy',[],'Formants',[],'Pitch',[],'VUVoiced'
,[],'Spectrum',[])

g.Options.Specgram =
struct('WindowType',[],'WindowWidth',[],'Overlap',[],'FftSize',[]','
FilterCoffecients',[],'DefaultColormap',[]);

g.Options.Energy =
struct('WindowType',[],'WindowWidth',[],'Overlap',[]);

g.Options.Formants =
struct('CofficientsNumber',[],'FftLength',[],'FormantsStart',[],'For
mantsEnd',[],'SegmentLength',[]);
```

```
g.Options.Pitch =
struct('PitchStart',[],'PitchEnd',[],'FilterType',[],'NumFiltPoles',
[],'FilterCuttoffFrequency',[],'SegmentLength',[],'WindowType',[]);

g.Options.Spectrum =
struct('LpcCoffNum',[],'FftLength',[],'WindowType',[],'SegmentLength
',[]);

g.Options.VUVoiced =
struct('eParameter',[],'pParameter',[],'rParameter',[],'PitchStart',
[],'PitchEnd',[],'FilterType',[],'NumFiltPoles',[],'FilterCuttoffFre
quency',[],'SegmentLength',[],'WindowType',[]);

%================================================================

SetDefaultsNew
```

======================*Defaults Setting*======================

```
function SetDefaultsNew
global g

%Spectrogram Options
g.Options.Specgram.WindowType = 'hamming';
g.Options.Specgram.WindowWidth = 128;
g.Options.Specgram.Overlap = 0.5;
g.Options.Specgram.FftSize = 1024;
g.Options.Specgram.FilterCoffecients = 0.9;
g.Options.Specgram.DefaultColormap = 'gray';

%================================================================

%Energy Options
g.Options.Energy.WindowType = 'hamming';
g.Options.Energy.WindowWidth = 220;
g.Options.Energy.Overlap = 0.5;

%================================================================

%Formants Options
g.Options.Formants.CofficientsNumber = 14;
g.Options.Formants.FftLength = 1024;
g.Options.Formants.FormantsStart = 150;
g.Options.Formants.FormantsEnd = 3400;
g.Options.Formants.SegmentLength = 220;

%================================================================

%Pitch Options
g.Options.Pitch.PitchStart = 0.0035;
g.Options.Pitch.PitchEnd = 0.02;
g.Options.Pitch.FilterType = 'butter';
g.Options.Pitch.NumFiltPoles = 6;
g.Options.Pitch.FilterCuttoffFrequency = 600;
g.Options.Pitch.SegmentLength = 0.0512;
g.Options.Pitch.WindowType = 'hamming';
g.Options.Pitch.Overlap = 0.5;

%================================================================
```

```
%Spectrum Options
g.Options.Spectrum.CofficientsNumber = 14;
g.Options.Spectrum.FftLength = 1024;
g.Options.Spectrum.WindowType = 'hamming';
g.Options.Spectrum.SegmentLength = 0.0256;

%================================================================

%VUVoiced Options
g.Options.VUVoiced.eParameter = 0.02;
g.Options.VUVoiced.pParameter = 0.285;
g.Options.VUVoiced.rParameter = 0.1425;
g.Options.VUVoiced.PitchStart = 0.0035;
g.Options.VUVoiced.PitchEnd = 0.02;
g.Options.VUVoiced.FilterType = 'butter';
g.Options.VUVoiced.NumFiltPoles = 6;
g.Options.VUVoiced.FilterCuttoffFrequency = 600;
g.Options.VUVoiced.SegmentLength = 0.0256;
g.Options.VUVoiced.WindowType = 'hamming';
g.Options.VUVoiced.Overlap = 0.5;

%================================================================
```

## ========== *Interface Object and Analysis Initialization* ==========

```
function FigureHandle = int_menu()
% This is the machine-generated representation of a Handle Graphics
object
% and its children.  Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was
saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your
path.
%
% NOTE: certain newer features in MATLAB may not have been saved in
this
% M-file due to limitations of this format, which has been
superseded by
% FIG-files.  Figures which have been annotated using the plot
editor tools
% are incompatible with the M-file/MAT-file format, and should be
saved as
% FIG-files.
%================================================================
global g;

Def_menuNew;
%================================================================
g.Figure.Dimensions=[750 520];
g.Figure.AspectRatio=0.75;
g.Figure.Margin=20;
%================================================================
```

```
g.Figure.Handle = figure('Color',[0.97 0.97 0.92], ...
   'FileName','C:\MATLABR11\project\try1.m', ...
   'PaperPosition',[18 180 576 432], ...
   'PaperUnits','points', ...
   'Position',[g.Figure.Margin g.Figure.Margin
g.Figure.Dimensions(1) g.Figure.Dimensions(2)], ...
   'Tag','Fig1', ...
   'Name','Speech Analysis',...
   'NumberTitle','off', ...
   'MenuBar','none',...
   'ToolBar','none');
%==========================================================
% Adding New Menu

% Menu No 1
g.Figure.Menu(1).Name='&File';
g.Figure.Menu(1).Function='';
g.Figure.Menu(1).Parent=g.Figure.Handle;
g.Figure.Menu(1).Enable='on';
% Menu No 2
g.Figure.Menu(2).Name='&Tool';
g.Figure.Menu(2).Function='';
g.Figure.Menu(2).Parent=g.Figure.Handle;
g.Figure.Menu(2).Enable='off';
% Menu No 3

g.Figure.Menu(3).Name='&Spectrogram';
g.Figure.Menu(3).Function='';
g.Figure.Menu(3).Parent=g.Figure.Handle;
g.Figure.Menu(3).Enable='off';

% Menu No 5
g.Figure.Menu(4).Name='&V/UV';
g.Figure.Menu(4).Function='V_UV2';
g.Figure.Menu(4).Parent=g.Figure.Handle;
g.Figure.Menu(4).Enable='off';
% Menu No 6
g.Figure.Menu(5).Name='&Pitch';
g.Figure.Menu(5).Function='';
g.Figure.Menu(5).Parent=g.Figure.Handle;
g.Figure.Menu(5).Enable='off';
% Menu No 7
g.Figure.Menu(6).Name='&Formants';
g.Figure.Menu(6).Function='';
g.Figure.Menu(6).Parent=g.Figure.Handle;
g.Figure.Menu(6).Enable='off';
% Menu No 8
g.Figure.Menu(7).Name='&Energy';
g.Figure.Menu(7).Function='Energy';
g.Figure.Menu(7).Parent=g.Figure.Handle;
g.Figure.Menu(7).Enable='off';
% Menu No 9
g.Figure.Menu(8).Name='&Spectrum';
g.Figure.Menu(8).Function='Spectrum';
g.Figure.Menu(8).Parent=g.Figure.Handle;
g.Figure.Menu(8).Enable='off';

% Menu No 9
g.Figure.Menu(9).Name='&Play';
g.Figure.Menu(9).Function='';
g.Figure.Menu(9).Parent=g.Figure.Handle;
```

```
g.Figure.Menu(9).Enable='off';

g.Figure.NumMenu=length(g.Figure.Menu);
%===========================================================
for i=1:g.Figure.NumMenu
   h(i)=uimenu('Parent',g.Figure.Menu(i).Parent,...
       'Label',g.Figure.Menu(i).Name,...
       'callback',g.Figure.Menu(i).Function,...
       'Enable',g.Figure.Menu(i).Enable,...
       'Tag',g.Figure.Menu(i).Name);
   g.Figure.Menu(i).Handle=h(i);
end
%===========================================================

% Adding SubMenu
% SubMenu 1
g.Figure.SubMenu(1).Name='&Averaged Pitch';
g.Figure.SubMenu(1).Function='Pitch(1)';
g.Figure.SubMenu(1).Parent=g.Figure.Menu(5).Handle;
% SubMenu 2
g.Figure.SubMenu(2).Name='&Single Pitch';
g.Figure.SubMenu(2).Function='Pitch(2)';
g.Figure.SubMenu(2).Parent=g.Figure.Menu(5).Handle;

g.Figure.SubMenu(3).Name='&Pitch Curve';
g.Figure.SubMenu(3).Function='PitchCurve';
g.Figure.SubMenu(3).Parent=g.Figure.Menu(5).Handle;

g.Figure.SubMenu(4).Name='&All Data'
g.Figure.SubMenu(4).Function='play_Data';
g.Figure.SubMenu(4).Parent=g.Figure.Menu(9).Handle;

g.Figure.SubMenu(5).Name='&Voiced Segment'
g.Figure.SubMenu(5).Function='play_voiced';
g.Figure.SubMenu(5).Parent=g.Figure.Menu(9).Handle;

g.Figure.SubMenu(6).Name='&Data Segment';
g.Figure.SubMenu(6).Function='PlaySeg';
g.Figure.SubMenu(6).Parent=g.Figure.Menu(9).Handle;

g.Figure.SubMenu(7).Name='&Spectrogram'
g.Figure.SubMenu(7).Function='SpegramNew';
g.Figure.SubMenu(7).Parent=g.Figure.Menu(3).Handle;

g.Figure.SubMenu(8).Name='&Color'
g.Figure.SubMenu(8).Function='';
g.Figure.SubMenu(8).Parent=g.Figure.Menu(3).Handle;

g.Figure.SubMenu(9).Name='&Zoom'
g.Figure.SubMenu(9).Function='Zooming';
g.Figure.SubMenu(9).Parent=g.Figure.Menu(2).Handle;

g.Figure.SubMenu(10).Name='&Reset Defaults';
g.Figure.SubMenu(10).Function='ResetDefaults';
g.Figure.SubMenu(10).Parent=g.Figure.Menu(2).Handle;

g.Figure.SubMenu(11).Name='&Option'
g.Figure.SubMenu(11).Function='';
g.Figure.SubMenu(11).Parent=g.Figure.Menu(2).Handle;

g.Figure.SubMenu(12).Name='&Open'
```

```
g.Figure.SubMenu(12).Function='Waveform';
g.Figure.SubMenu(12).Parent=g.Figure.Menu(1).Handle;

g.Figure.SubMenu(13).Name='&New'
g.Figure.SubMenu(13).Function='Record';
g.Figure.SubMenu(13).Parent=g.Figure.Menu(1).Handle;

g.Figure.SubMenu(14).Name='&Exit'
g.Figure.SubMenu(14).Function='close(gcf)';
g.Figure.SubMenu(14).Parent=g.Figure.Menu(1).Handle;

g.Figure.SubMenu(15).Name='&Formants'
g.Figure.SubMenu(15).Function='Formants';
g.Figure.SubMenu(15).Parent=g.Figure.Menu(6).Handle;

g.Figure.SubMenu(16).Name='&Formants History'
g.Figure.SubMenu(16).Function='FormantsHistory';
g.Figure.SubMenu(16).Parent=g.Figure.Menu(6).Handle;

g.Figure.NumSubMenu=length(g.Figure.SubMenu);
%============================================================
for i=1:g.Figure.NumSubMenu
   h(i)=uimenu('Parent',g.Figure.SubMenu(i).Parent,...
      'Label',g.Figure.SubMenu(i).Name,...
      'callback',g.Figure.SubMenu(i).Function,...
      'Tag',g.Figure.SubMenu(i).Name);
   g.Figure.SubMenu(i).Handle=h(i);
end
%============================================================

% Adding SSubMenu
% SSubMenu 1
g.Figure.SSubMenu(1).Name='&gray'
g.Figure.SSubMenu(1).Function='colormap(abs(gray-1))';
g.Figure.SSubMenu(1).Parent=g.Figure.SubMenu(8).Handle;

% SSubMenu 2
g.Figure.SSubMenu(2).Name='&hsv'
g.Figure.SSubMenu(2).Function='colormap(hsv)';
g.Figure.SSubMenu(2).Parent=g.Figure.SubMenu(8).Handle;

% SSubMenu 3
g.Figure.SSubMenu(3).Name='&hot'
g.Figure.SSubMenu(3).Function='colormap(hot)';
g.Figure.SSubMenu(3).Parent=g.Figure.SubMenu(8).Handle;

% SSubMenu 4
g.Figure.SSubMenu(4).Name='&cool'
g.Figure.SSubMenu(4).Function='colormap(cool)';
g.Figure.SSubMenu(4).Parent=g.Figure.SubMenu(8).Handle;

% SSubMenu 5
g.Figure.SSubMenu(5).Name='&jet'
g.Figure.SSubMenu(5).Function='colormap(jet)';
g.Figure.SSubMenu(5).Parent=g.Figure.SubMenu(8).Handle;

% SSubMenu 6
g.Figure.SSubMenu(6).Name='&Spectrogram'
g.Figure.SSubMenu(6).Function='OptionSpecN';
g.Figure.SSubMenu(6).Parent=g.Figure.SubMenu(11).Handle;
```

```
% SSubMenu 7
g.Figure.SSubMenu(7).Name='&Energy'
g.Figure.SSubMenu(7).Function='EnergyOptions';
g.Figure.SSubMenu(7).Parent=g.Figure.SubMenu(11).Handle;

% SSubMenu 8
g.Figure.SSubMenu(8).Name='&Formants'
g.Figure.SSubMenu(8).Function='FormantsOptions';
g.Figure.SSubMenu(8).Parent=g.Figure.SubMenu(11).Handle;

% SSubMenu 8
g.Figure.SSubMenu(9).Name='&Pitch'
g.Figure.SSubMenu(9).Function='PitchOptions';
g.Figure.SSubMenu(9).Parent=g.Figure.SubMenu(11).Handle;

% SSubMenu 8
g.Figure.SSubMenu(10).Name='&Spectrum'
g.Figure.SSubMenu(10).Function='SpectrumOptions';
g.Figure.SSubMenu(10).Parent=g.Figure.SubMenu(11).Handle;

% SSubMenu 8
g.Figure.SSubMenu(11).Name='&V/Uvoiced'
g.Figure.SSubMenu(11).Function='VUVoicedOptions';
g.Figure.SSubMenu(11).Parent=g.Figure.SubMenu(11).Handle;

g.Figure.NumSSubMenu=length(g.Figure.SSubMenu);

%=========================================================
for i=1:g.Figure.NumSSubMenu
   h(i)=uimenu('Parent',g.Figure.SSubMenu(i).Parent,...
      'Label',g.Figure.SSubMenu(i).Name,...
      'callback',g.Figure.SSubMenu(i).Function,...
      'Tag',g.Figure.SSubMenu(i).Name);
   g.Figure.SSubMenu(i).Handle=h(i);
end
%=========================================================

% Buttons initialization

%for i=1:g.Figure.NumFunction,
%  h(i) = uicontrol('Parent',g.Figure.Handle, ...
%        'Units','normalized', ...
%        'FontUnits','points', ...
%        'BackgroundColor',[0.75 0.75 0.75], ...
%        'FontSize',10, ...
%        'ListboxTop',0, ...
%        'Position',[0.8547 0.056*i 0.120 0.0485], ...
%         'Callback',g.Button(i).Function, ...
%        'String',g.Button(i).Name, ...
%        'Tag',g.Button(i).Name);
%   g.Button(i).Handle=h(i);
%end

%=============================================================

h1 = axes('Parent',g.Figure.Handle, ...
   'Units','normalized', ...
   'Color',[1 1 1], ...
   'Position',[0.15 0.6 0.68 0.3], ...
   'Box','on',...
   'Tag','Axes1');
```

```
h1 = axes('Parent',g.Figure.Handle, ...
    'Units','normalized', ...
    'Color',[1 1 1], ...
    'Position',[0.15 0.15 0.68 0.3], ...
    'Box','on',...
    'Tag','Axes2');

%=============================================================

g.Figure.Axes1Handle = findobj('Tag','Axes1')
g.Figure.Axes2Handle = findobj('Tag','Axes2')

set(gcf,'ResizeFcn','rsz');
```

## ==================== *Waveform display* ====================

```
function Waveform
% This function give the user the ability to choose the wave file
% he wants, then display the waveform of that sound

global g;

% Signal is the name of our data (wave data)
[filename,pathname]=uigetfile('*.wav','Selec the sound wave file')
% This function display the dialog box then give us the file name
% of the selectd file
[g.signal.Data,g.signal.fs]=wavread(strcat(pathname,filename));
% Read the wave which has the name (filename)
axes(g.Figure.Axes1Handle);
set(g.Figure.Axes1Handle,'UserData',g.signal.Data);
t=(1:length(g.signal.Data))/g.signal.fs;
plot(t,g.signal.Data,'color',[0 0 0]);
% display the waveform on the interface


xlabel('Time,
Seconds','FontSize',10,'FontUnit','normalized','FontWeight','normal'
);
ylabel('Amplitude, m
volt','FontSize',10,'FontUnit','normalized','FontWeight','normal');
title(filename,'FontSize',10,'FontUnit','normalized','FontWeight','n
ormal');

for i=1:g.Figure.NumMenu
    set(g.Figure.Menu(i).Handle,'Enable','on')
end

axes(g.Figure.Axes2Handle);
cla;
```

=============================================================

## ================*Energy Analysis* ========================

```
function Energy
% This Function make the classification of voiced and
ung.signal.voiced speech
% and return with voiced frames

global g;
```

```
%WindowLength=256;

N =
floor(length(g.signal.Data)/(g.Options.Energy.WindowWidth*g.Options.
Energy.Overlap));

%using hamming window 256 points
window = feval(g.Options.Energy.WindowType,
g.Options.Energy.WindowWidth);

j=1;
Energy=[];

for i=1:N-1
   segment = g.signal.Data(j:g.Options.Energy.WindowWidth-1+j);
   %take a segment of the data with 256 length

   windowedseg = sum((segment.*window).^2);

   Energy = [Energy,windowedseg'];
   j = j + g.Options.Energy.WindowWidth * g.Options.Energy.Overlap;
end

Energy = Energy(:);
axes(g.Figure.Axes2Handle)
t=(1:N-
1)/g.signal.fs*g.Options.Energy.WindowWidth*g.Options.Energy.Overlap
plot(t,Energy,'color',[0 0 0]);

xlabel('Time in
second','FontSize',10,'FontUnit','normalized','FontWeight','normal')
;
ylabel('Energy in
what','FontSize',10,'FontUnit','normalized','FontWeight','normal');
title('ST-Energy
Curve','FontSize',10,'FontUnit','normalized','FontWeight','normal');
```

---

```
function fig = EnergyOptions()
% This is the machine-generated representation of a Handle Graphics
object
% and its children.  Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was
saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your
path.
%
% NOTE: certain newer features in MATLAB may not have been saved in
this
% M-file due to limitations of this format, which has been
superseded by
% FIG-files.  Figures which have been annotated using the plot
editor tools
```

```
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.

load EnergyOptions

h0 = figure('Color',[0.97 0.97 0.92], ...
   'Colormap',mat0, ...
   'FileName','C:\MATLABR11\project\EnergyOption.m', ...
   'MenuBar','none', ...
   'Name','Energy Options', ...
   'NumberTitle','off', ...
   'PaperPosition',[18 180 500 432], ...
   'PaperUnits','points', ...
   'Position',[230 230 304 130], ...
   'Tag','Fig1', ...
   'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[0.97 0.97 0.95], ...
   'Callback','EnergyWindowType', ...
   'ListboxTop',0, ...
   'Position',[0.6611842105263156 0.5615384615384615
0.2434210526315789 0.1307692307692308], ...
   'String',['hamming ';'hanning ';'kaiser
';'bartlett';'blackman'], ...
   'Style','popupmenu', ...
   'Tag','PopupMenu1', ...
   'Value',1);
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[0.97 0.97 0.92], ...
   'FontAngle','italic', ...
   'FontWeight','demi', ...
   'ListboxTop',0, ...
   'Position',[0.01973684210526316 0.7692307692307693
0.3355263157894737 0.1461538461538462], ...
   'String','Window Length', ...
   'Style','text', ...
   'Tag','Text1');
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[0.97 0.97 0.92], ...
   'FontAngle','italic', ...
   'FontWeight','demi', ...
   'ListboxTop',0, ...
   'Position',mat1, ...
   'String','Overlab', ...
   'Style','text', ...
   'Tag','Text4');
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
   'Callback','Apply', ...
   'ListboxTop',0, ...
   'Position',[0.1513157894736842 0.1461538461538462
0.2335526315789473 0.1769230769230769], ...
   'String','Apply', ...
   'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
```

```
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.92], ...
    'FontAngle','italic', ...
    'FontWeight','demi', ...
    'ListboxTop',0, ...
    'Position',mat2, ...
    'String','Window Type', ...
    'Style','text', ...
    'Tag','Text4');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','close(gcf)', ...
    'ListboxTop',0, ...
    'Position',[0.5592105263157895 0.1384615384615385
0.2138157894736842 0.1846153846153846], ...
    'String','Cancel', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.95], ...
    'Callback','EnergyWindowWidthSeting', ...
    'ListboxTop',0, ...
    'Position',[0.08552631578947366 0.5230769230769231
0.1842105263157895 0.1692307692307692], ...
    'Style','edit', ...
    'Tag','Edit1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.95], ...
    'Callback','EnergyOverlabSeting', ...
    'ListboxTop',0, ...
    'Position',[0.3782894736842105 0.5153846153846154 0.1875
0.1769230769230769], ...
    'Style','edit', ...
    'Tag','Edit2');
if nargout > 0, fig = h0; end
```

===============================================================

## *Overlab Setting*

```
function EnergyOverlabSeting
global g;
Edit2Handle = findobj('Tag','Edit2');
O = get(Edit2Handle,'String');
g.Options.Energy.Overlap = eval(O);
```

===============================================================

## *Window Type Setting*

```
function EnergyWindowType
global g;
h=findobj('Tag','PopupMenu1');
val = get(h,'Value');

close(gcf)
if val == 1
   g.Options.Energy.WindowType = 'hamming'
elseif val == 2
   g.Options.Energy.WindowType = 'hanning'
```

```
elseif val == 3
   g.Options.Energy.WindowType = 'kaiser'
elseif val == 4
   g.Options.Energy.WindowType = 'bartlett'
elseif val == 5
   g.Options.Energy.WindowType = 'blackman'
end
```

==================================================================

## *Window Width Setting*

```
function EnergyWindowWidthSeting
global g;
Edit2Handle = findobj('Tag','Edit1');
W=get(Edit2Handle,'String');
g.Options.Energy.WindowWidth = eval(W);
```

==================================================================

## ==================*Formant Analysis*====================

```
function Peak=FindPeaks(H)

% finding the sign of the derivative
d = sign(diff(H));
Peak = [];
% add the adjacent element to see the point of converting form + to
- and vice versa
Peak = diff(d);

% finding the peaks indexes
Peak = find(Peak==-2);
```

==================================================================

```
function Formants=Formants
global g;

EST = [320 1440 2760 3200];    % EST in Hz
EST = fix(EST*g.Options.Formants.FftLength/g.signal.fs);   % EST in
index
Term = ginput(1);
Term = fix(Term*g.signal.fs);
segment =
g.signal.Data(Term(1):(Term(1)+g.Options.Formants.SegmentLength));

S = [0 0 0 0];        % Four slots
%=================================================================

a = lpc(segment,g.Options.Formants.CofficientsNumber);
% Vocal tract Transfer Function at r=1
Horignal = 1./abs(fft(a,g.Options.Formants.FftLength));

r = 1;
[S,Peak,H] = lpcFormants(a,r,g.signal.fs,S,EST);
%=================================================================

% step 5 deal with Unfilled Slots
% if S1, S2, S3 are all filled, go to Setp 6. Otherwise:
% Recompute the spectrum on a circle with radius less than one to
enhance the formants
```

```
if ~isempty(find(S(1:3)==0)),

   while 1,
      r = r-0.004;
      if r<0.88, break; end;
      [S, Peak,H] = lpcFormants(a,r,g.signal.fs,S,EST);
      if isempty(find(S(1:3)==0)), break; end;
      %plot(H);
   end;

end;

%if ~isempty(S(1:3)==0),

%    while 1,
%      r = r-0.004;
%       if r<0.88, break; end;
%      [S, Peak,H] = lpcFormants(a,r,g.signal.fs,S,EST);
%      if isempty(S(1:3)==0), break; end;
      %plot(H);
%   end;

%end;

%if the empty slot was S3, and enhancement failed to yield a peak,
%then the peak in S4 is moved down to S3, assuming that F3 was
mistakenly called F4
if S(3) == 0
   S(3) = S(4);
end

%whether or not enhancement has succeeded, the amplitudes of the
peaks are reset to
%the amplitudes in the original spectrum.
k = find(S~=0);
PeakAmp = [];
for i = 1:length(k)
   PeakAmp(i) = Horignal(S(k(i)));
end

%if a slot is empty, keep the original formant estimate for that
formant
for i = 1:length(S)
   if S(i)==0
      S(i)=EST(i);
   end
end

%=================================================================
Formants = S /g.Options.Formants.FftLength * g.signal.fs;

axes(g.Figure.Axes2Handle);
for i=1:length(Peak)
   x(i)=Horignal(Peak(i))
end
plot(Peak,x,'+')
hold on
plot(Horignal(1:end/2),'color','k');
hold off
```

```
msgbox(['F1 = ',num2str(Formants(1)),' Hz','      F2 = ',
num2str(Formants(2)), 'Hz','      F3 = ', num2str(Formants(3)),
'Hz'],'Formants in Hz','help');
```

---

```
function [S, Peak,H]=lpcFormants(a,r,fs,S,EST)
global g;

% r contribute in Peak enhanciment
a = a.*r.^(-(0:g.Options.Formants.CofficientsNumber));
% Vocal tract Trasfer Function
H = 1./abs(fft(a,g.Options.Formants.FftLength));

% Discard Formants more than g.Options.Formants.FormantsEnd 3400 Hz
ending = fix((g.Options.Formants.FormantsEnd-
1)*g.Options.Formants.FftLength/fs)+1;
begining = fix((g.Options.Formants.FormantsStart-
1)*g.Options.Formants.FftLength/fs)+1;  % check +-1, I check
H = H(begining:ending);
%=====================================================================
% Step 1 Fetch Peaks. Find the frequencies and amplitude of up to
four peaks
% in the region form 150 to 3400 Hz.
Peak = FindPeaks(H);
Peak = Peak(1:min([4, length(Peak)]));
Peak = Peak + begining; % add the begining to the peak index
NumPeak = length(Peak);
%=====================================================================

Th=filter([1/2 1/2],1,EST);
Th=Th(2:end);
%=====================================================================
% Step 2 Fill each formant slot form 1 to 4, with the best candidate
peak Peak(j),
% by the following rule: The peak Peak(j) closest in frequency to
estimate
% EST(i) goes into slot S(i).
S(sum(kron(ones(3,1),Peak(:)')>kron(ones(1,NumPeak),Th(:)),1)+1)=Pea
k;

%=====================================================================

% step 4 deal with Unassigned Peaks

% Search for unassigned Peak
NumPeak=length(Peak);
D=kron(ones(1,NumPeak),S(:))- kron(ones(4,1),Peak(:)');
D=(D==0);
D=sum(D);
k=find(D==0);

%=====================================================================

% if there is a peak Peak(j=k) unassigned, and an S(i=k) unfilled,
fill the
% slot with the peak and go to step 5.
for i=1:length(k)

   if S(k(i))==0,
      S(k(i)) = Peak(k(i));
```

```
    % if there is a peak Peak(j=k) unassigned, but slot S(i=k) is
already filled,
    % check the amplitude of Peak(k) as follows:
    % if amp (Peak(k)) < 1/2 amp (peak already assigned to S(k)),
    % throw Peak(k) and go to step 5
    elseif (H(Peak(k(i))) >= 1/2*H(S(k(i))))
        if k(i)<4 & S(k(i)+1)==0,
            S(k(i)+1)=S(k(i));
            S(k(i))=Peak(k(i));
        elseif k(i)>1 & S(k(i)-1)==0,
            S(k(i)-1)=S(k(i));
            S(k(i))=Peak(k(i));
        end;
    end;
end;

%===================================================================
```

*Coefficient Number Setting*

```
function FormantsCoffNumSeting
global g;
Edit1Handle = findobj('Tag','Edit1');
C = get(Edit1Handle,'String');
g.Options.Formants.CofficientsNumber = eval(C);
```

*Formant End Setting*

```
function FormantsEndSeting
global g;
Edit4Handle = findobj('Tag','Edit4');
E = get(Edit4Handle,'String');
g.Options.Formants.FormantsEnd = eval(E);
```

*Formants Fft Length Setting*

```
function FormantsFftLengthSeting
global g;
Edit2Handle = findobj('Tag','Edit2');
F = get(Edit2Handle,'String');
g.Options.Formants.FftLength = eval(F);
```

*Formants Segment Length Setting*

```
function FormantsSegmentLengthSeting
global g;
Edit5Handle = findobj('Tag','Edit5');
S = get(Edit5Handle,'String');
g.Options.Formants.SegmentLength = eval(S);
```

*Formants Start Setting*

```
function FormantsStartSeting
global g;
Edit3Handle = findobj('Tag','Edit3');
S=get(Edit3Handle,'String');
g.Options.Formants.FormantsStart = eval(S);
```

---

*Formant Options Interface*

```
function fig = FormantsOptions()
% This is the machine-generated representation of a Handle Graphics
object
```

```
% and its children.  Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was
saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your
path.
%
% NOTE: certain newer features in MATLAB may not have been saved in
this
% M-file due to limitations of this format, which has been
superseded by
% FIG-files.  Figures which have been annotated using the plot
editor tools
% are incompatible with the M-file/MAT-file format, and should be
saved as
% FIG-files.

load FormantsOptions

h0 = figure('Color',[0.97 0.97 0.92], ...
    'Colormap',mat0, ...
    'FileName','C:\MATLABR11\work\project\FormantsOptions.m', ...
    'MenuBar','none', ...
    'Name','Formants Options', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 400 432], ...
    'PaperUnits','points', ...
    'Position',[160 230 458 130], ...
    'Tag','Fig1', ...
    'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.92], ...
    'FontAngle','italic', ...
    'FontWeight','demi', ...
    'ListboxTop',0, ...
    'Position',[0.04148471615720523 0.7153846153846154
0.1659388646288209 0.2307692307692308], ...
    'String','Cofficients Number', ...
    'Style','text', ...
    'Tag','Text1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.92], ...
    'FontAngle','italic', ...
    'FontWeight','demi', ...
    'ListboxTop',0, ...
    'Position',[0.2510917030567685 0.6923076923076923
0.1397379912663755 0.2538461538461538], ...
    'String','FFT Length', ...
    'Style','text', ...
    'Tag','Text4');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.92], ...
```

```
    'FontAngle','italic', ...
    'FontWeight','demi', ...
    'ListboxTop',0, ...
    'Position',[0.4344978165938864 0.7076923076923077
0.1593886462882096 0.2384615384615385], ...
    'String','Formants Start', ...
    'Style','text', ...
    'Tag','Text4');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.92], ...
    'FontAngle','italic', ...
    'FontWeight','demi', ...
    'ListboxTop',0, ...
    'Position',mat1, ...
    'String','Formants End', ...
    'Style','text', ...
    'Tag','Text4');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.92], ...
    'FontAngle','italic', ...
    'FontWeight','demi', ...
    'ListboxTop',0, ...
    'Position',mat7, ...
    'String','Segment Length', ...
    'Style','text', ...
    'Tag','Text4');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.95], ...
    'Callback','FormantsCoffNumSeting', ...
    'ListboxTop',0, ...
    'Position',mat2, ...
    'Style','edit', ...
    'Tag','Edit1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.95], ...
    'Callback','FormantsFftLengthSeting', ...
    'ListboxTop',0, ...
    'Position',mat3, ...
    'Style','edit', ...
    'Tag','Edit2');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.95], ...
    'Callback','FormantsStartSeting', ...
    'ListboxTop',0, ...
    'Position',mat4, ...
    'Style','edit', ...
    'Tag','Edit3');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.95], ...
    'Callback','FormantsEndSeting', ...
    'ListboxTop',0, ...
    'Position',mat5, ...
    'Style','edit', ...
    'Tag','Edit4');
h1 = uicontrol('Parent',h0, ...
```

```
    'Units','normalized', ...
    'BackgroundColor',[0.97 0.97 0.95], ...
    'Callback','FormantsSegmentLengthSeting', ...
    'ListboxTop',0, ...
    'Position',mat6, ...
    'Style','edit', ...
    'Tag','Edit5');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','Apply', ...
    'ListboxTop',0, ...
    'Position',[0.2358078602620087 0.07692307692307693
0.1506550218340611 0.1769230769230769], ...
    'String','Apply', ...
    'Tag','button1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','close(gcf)', ...
    'ListboxTop',0, ...
    'Position',[0.6193595342066958 0.07692307692307693
0.1506550218340611 0.1846153846153846], ...
    'String','Cancel', ...
    'Tag','button2');
if nargout > 0, fig = h0; end

===================================================================

function Formants=FormantsHistory
global g;

spegramNew;
hold on;
N =
floor(length(g.signal.Data)/((g.Options.Energy.WindowWidth*g.Options
.Energy.Overlap)+1));

window = feval(g.Options.Energy.WindowType,
g.Options.Energy.WindowWidth);

% This is tell the user that he should wait for porcessing
h = waitbar(0,'Please wait Formants History in progress');

j=1;
F=[];

for i=1:N
   segment = g.signal.Data(j:g.Options.Energy.WindowWidth-1+j);
   %take a segment of the data with 256 length
   windowedseg = segment.*window;

EST = [320 1440 2760 3200];   % EST in Hz
EST = fix(EST*g.Options.Formants.FftLength/g.signal.fs);  % EST in
index
segment = windowedseg;

S = [0 0 0 0];        % Four slots
%==================================================================
```

```
a = lpc(segment,g.Options.Formants.CofficientsNumber);
% Vocal tract Transfer Function at r=1
H = 1./abs(fft(a,g.Options.Formants.FftLength));
r = 1;
S = lpcFormants(a,r,g.signal.fs,S,EST);

%=================================================================

% step 5 deal with Unfilled Slots
% if S1, S2, S3 are all filled, go to Setp 6. Otherwise:
% Recompute the spectrum on a circle with radius less than one to
enhace the formants

if ~isempty(S(1:3)==0),

   while 1,
      r = r-0.004;
      if r<0.88, break; end;
      [S, Peak] = lpcFormants(a,r,g.signal.fs,S,EST);
      if isempty(S(1:3)==0), break; end;
   end;

end;
   %this is user wait indication
   waitbar(i/(N-1))

%if the empty slot was S3, and enhancement failed to yield a peak,
%then the peak in S4 is moved down to S3, assuming that F3 was
mistakenly called F4
if S(3) == 0
   S(3) = S(4);
end

%whether or not enhancement has succeeded, the amplitudes of the
peaks are reset to
%the amplitudes in the original spectrum.
k = find(S~=0);
PeakAmp = [];
for i = 1:length(k)
   PeakAmp(i) = H(S(k(i)));
end

%if a slot is empty, keep the original formant estimate for that
formant
for i = 1:length(S)
  if S(i)==0
      S(i)=EST(i);

  end
end

%=================================================================

Formants = S /g.Options.Formants.FftLength * g.signal.fs;

F = [F,Formants];
   j = j + g.Options.Energy.WindowWidth * g.Options.Energy.Overlap;
end
close(h)
```

```
axes(g.Figure.Axes2Handle);
F = reshape(F,4,length(F)/4);
F(4,:)=[];
x=(1:N)/g.signal.fs * g.Options.Energy.WindowWidth *
g.Options.Energy.Overlap;
plot(x,F,'.','color','w');
title('Formants
History','FontSize',10,'FontUnit','normalized','FontWeight','normal'
);
hold off
```

## ============== *Voiced / Unvoiced Classification* ============

```
function V_Unvoiced
% This Function make the classification of voiced and
ung.signal.voiced speech
% and return with voiced frames
global g

fs = g.signal.fs;
[B,A] =
feval(g.Options.Pitch.FilterType,g.Options.Pitch.NumFiltPoles,g.Opti
ons.Pitch.FilterCuttoffFrequency/(fs/2));
%low pass filter with cutoff frequecy at 600 Hz
filtered = filter(B,A,g.signal.Data);
%each segment has length of 51.2ms (Term is define that).
Term = fix(g.Options.VUVoiced.SegmentLength * fs);
%Number of frames per or segments per the input sound wave.
N = floor(length(g.signal.Data)/(Term/2));

j=1;
c=[];
NumVoicFram = 0;
g.signal.voiced = [];
% This is tell the user that he should wait for porcessing
h = waitbar(0,'Please wait...');
%lmin to lmax represent the segment from 3 to 20 ms
lmin=fix(g.Options.VUVoiced.PitchStart * fs);
lmax=fix(g.Options.VUVoiced.PitchEnd * fs);

for i=1:N-1

    %take a segment of the data with 51.2 ms length
    segment=g.signal.Data(j:Term+j);
    %this is user wait indication
    waitbar(i/(N-1))

    correlatedsegment=xcorr(segment);

    R = correlatedsegment;
    M = length(segment);


    R=R(M:end);
    R=R(1:min([lmax+1 M]));
    MaxValue=max(R(lmin:end));
    K=min(find(R(lmin:end)==MaxValue))+lmin-1;
    e=sqrt(R(1)/Term);
    p=R(K)/R(1);
    r=sqrt(sum((R(lmin:end)/R(1)).^2)/length(R(lmin:end)));

    %define thersholds e, p, and r
```

```
    if (e > g.Options.VUVoiced.eParameter) & (p >
g.Options.VUVoiced.pParameter) & (r > g.Options.VUVoiced.rParameter)
        c0=1;
        NumVoicFram=NumVoicFram+1;
        g.signal.voiced=[g.signal.voiced,segment'];
    else
        c0=0;
    end
    c=[c c0];
    j=j+fix(Term/2);
end
close(h)

c=c;
%this is for plotting the voiced and unvoiced frames with deferent
colors
g.signal.voiced=g.signal.voiced(:);

Ones128 = ones(1,Term/2);

Mull = kron(c,Ones128);

Signal = g.signal.Data(1:length(Mull));
Signal = Signal';
VoicedOnly = Mull.*Signal;

axes(g.Figure.Axes1Handle);
t=(1:length(VoicedOnly))/g.signal.fs;
plot(t,VoicedOnly,'color','r');

CMull = abs(Mull-1);
% Complement of Mull
hold on

UnVoicedOnly = CMull.*Signal;

plot(t,UnVoicedOnly,'color','k')

hold off
title('Red part represent the voiced
speech','FontSize',10,'FontUnit','normalized','FontWeight','normal')
;
xlabel('Time in
second','FontSize',10,'FontUnit','normalized','FontWeight','normal')
;
ylabel('Amplitude in m
volt','FontSize',10,'FontUnit','normalized','FontWeight','normal');

===================================================================
```

*VUV Cut off Frequency Setting*

```
function VUVCuttoffSeting
global g;
Edit8Handle = findobj('Tag','Edit8');
C = get(Edit8Handle,'String');
g.Options.VUVoiced.FilterCuttoffFrequency = eval(C);
```

*VUV e Parameter Setting*

```
function VUVeParameterSeting
global g;
```

```
Edit1Handle = findobj('Tag','Edit1');
e = get(Edit1Handle,'String');
g.Options.VUVoiced.eParameter = eval(e);
```

*VUV Filter Type Setting*

```
function VUVFilterTypeSeting
global g;
Popupmenu1Handle = findobj('Tag','PopupMenu1');
val = get(Popupmenu1Handle,'Value');
%close(gcf)
if val == 1
   g.Options.VUVoiced.FilterType = 'butter'
elseif val == 2
   g.Options.VUVoiced.FilterType = 'buttord'
elseif val == 3
   g.Options.VUVoiced.FilterType = 'besself'
elseif val == 4
   g.Options.VUVoiced.FilterType = 'chepy1'
elseif val == 5
   g.Options.VUVoiced.FilterType = 'ellip'
end
```

*VUV Number of Poles Setting*

```
function VUVNPolesSeting
global g;
Edit7Handle = findobj('Tag','Edit7');
P = get(Edit7Handle,'String');
g.Options.VUVoiced.NumFiltPoles = eval(P);
```

*VUV Window Overlab Setting*

```
function VUVOverlabSeting
global g;
Edit9Handle = findobj('Tag','Edit9');
O = get(Edit9Handle,'String');
g.Options.VUVoiced.Overlap = eval(O);
```

*VUV Pitch End Setting*

```
function VUVPitchEndSeting
global g;
Edit5Handle = findobj('Tag','Edit5');
E = get(Edit5Handle,'String');
g.Options.VUVoiced.PitchEnd = eval(E);
```

*VUV Pitch Start Setting*

```
function VUVPitchStartSeting
global g;
Edit4Handle = findobj('Tag','Edit4');
S = get(Edit4Handle,'String');
g.Options.VUVoiced.PitchStart = eval(S);
```

*VUV p Parameter Setting*

```
function VUVpParameterSeting
global g;
Edit2Handle = findobj('Tag','Edit2');
p = get(Edit2Handle,'String');
g.Options.VUVoiced.pParameter = eval(p);
```

*VUV r Parameter Setting*

```
function VUVrParameterSeting
```

```
global g;
Edit3Handle = findobj('Tag','Edit3');
r = get(Edit3Handle,'String');
g.Options.VUVoiced.rParameter = eval(r);
```

*VUV Segment Length Setting*

```
function VUVSegmentLengthSeting
global g;
Edit6Handle = findobj('Tag','Edit6');
L = get(Edit6Handle,'String');
g.Options.VUVoiced.SegmentLength = eval(L);
```

*VUV Window Type Setting*

```
function VUVWindowTypeSeting
global g;
Popupmenu2Handle = findobj('Tag','PopupMenu2');
val = get(Popupmenu2Handle,'Value');
%close(gcf)
if val == 1
   g.Options.VUVoiced.WindowType = 'hamming'
elseif val == 2
   g.Options.VUVoiced.WindowType = 'hanning'
elseif val == 3
   g.Options.VUVoiced.WindowType = 'kaiser'
elseif val == 4
   g.Options.VUVoiced.WindowType = 'bartlett'
elseif val == 5
   g.Options.VUVoiced.WindowType = 'blackman'
end
```

==================================================================

====================*Pitch Analysis*=====================

```
function Pitch(Position)

switch Position
   case 1
      Pitch1Average
   case 2
      Pitch1
end

function Pitch1
global g;
%Select 51.2ms of the input sound wave
fs = g.signal.fs;
Term = ginput(1);
Term = fix(Term*fs);
begining = max(1,Term(1));
segmentlength = fix(g.Options.Pitch.SegmentLength*fs);
ending = min(length(g.signal.Data),Term(1)+segmentlength);
selected = g.signal.Data(begining:ending);
%Filter the  selected segment with a butterworth filter having a
cutoff frequency of 600 Hz
[B,A] = butter(6,600/(fs/2));
filtered = filter(B,A,selected);
window = feval(g.Options.Pitch.WindowType,length(filtered));

%==================================================================
```

```
%Compute the autocorrelation function for the selected segment, then
find the maximum value
% inside the pitch range from 30 to 200 ms.
%The index of this value will be the lag (L) after adding lmin.
segment = filtered.*window;

R = xcorr(segment);
M = length(segment);

lmin = fix(g.Options.Pitch.PitchStart*fs);
lmax = fix(g.Options.Pitch.PitchEnd*fs);

R = R(M:end);
RR = R(max(lmin, 1):min(lmax, M));

MaxValue = max(RR);

Lag = min(find(RR==MaxValue));
Lag = Lag+lmin;
L = Lag;

%==================================================================

%This part checks for peaks at one-half, one-third, one-fourth, one-
fifth,
% and one-sixth of the first estimate of the pitch period.
% If L/2 (rounded up) is within th pitch range, the maximum value of
the autocorrelation
% within (L/2)-5 to (L/2)+5 is located. If (L/2)-5 is less than
lmin, lmin is chosen as the
% lower limit instead of (L/2)-5. If the new peak is greater than
one-half of the old peak,
% the new corresponding lag replaces the old corresponding lag L.

if (lmin <= round(L/2))&(round(L/2)<=lmax)
   seg = R(fix(max(lmin,L/2-5)):fix(L/2+5));
   peak = max(seg);
   %check for pitch period doubling error
   if peak > (1/2)*R(L)
      L = L/2;
      %check for pitch period double doubling error (fourfold
errors)
      if (lmin <= round(L/2))&(round(L/2)<=lmax)
            seg = R(fix(max(lmin,L/2-5)):fix(L/2+5));
            peak = max(seg);
            if peak > (1/2)*R(L)
               L = L/2;
               break
            end
      end
      %check for pitch period errors of sixfold
      if (lmin <= round(L/3))&(round(L/3)<=lmax)
            seg = R(fix(max(lmin,L/3-5)):fix(L/3+5));
            peak = max(seg);
            if peak > (1/2)*R(L)
               L = L/3;
               break
            end
      end

   end
```

```
   %check for pitch period tripling errors
   if (lmin <= round(L/3))&(round(L/3)<=lmax)
         seg = R(fix(max(lmin,L/3-5)):fix(L/3+5));
         peak = max(seg);
         if peak > (1/2)*R(L)
            L = L/3;
            break
         end
   end
   %check for pitch period errors of fivefold
   if (lmin <= round(L/5))&(round(L/5)<=lmax)
         seg = R(fix(max(lmin,L/5-5)):fix(L/5+5));
         peak = max(seg);
         if peak > (1/2)*R(L)
            L = L/5;
            break
         end
   end

end
%=================================================================

%The pitch period calculated through the following formula.
pitch = fs/L;

%=================================================================
axes(g.Figure.Axes2Handle)
plot(L,R(L),'+')
hold on
plot(R,'color',[0 0 0]);
hold off
msgbox(['Pitch frequency is ',num2str(fs/L),' Hz'],'Pitch Frequcney
in Hz','help');


function Pitch1Average
global g;
%select a voiced area from the speech waveform.
fs = g.signal.fs;
Term = ginput(2);
begining = max(1,fix(Term(1)*fs));
ending = min(length(g.signal.Data),fix(Term(2)*fs));
selected = g.signal.Data(begining:ending);
%divide this area to an equal segments each segment has the length
of 51.2 ms.
N =
floor(length(selected)/(g.Options.Pitch.SegmentLength*g.Options.Pitc
h.Overlap*fs));

%Filter the  selected segment with a butterworth filter having a
cutoff frequency of 600 Hz
[B,A] =
feval(g.Options.Pitch.FilterType,g.Options.Pitch.NumFiltPoles,g.Opti
ons.Pitch.FilterCuttoffFrequency/(fs/2));
filtered = filter(B,A,selected);

%calculate pitch lag for each segment then take the average
j=1;
LL=[];
```

```
window =
feval(g.Options.Pitch.WindowType,round(g.Options.Pitch.SegmentLength
*fs));

for i = 1:N-1
   %Compute the autocorrelation function for the selected segment,
then find the maximum value
   %inside the pitch range from 30 to 200 ms.
   %The index of this value will be the lag (L) after adding lmin.
   segment = filtered(j:round(g.Options.Pitch.SegmentLength*fs)+j-
1).*window;

   R = xcorr(segment);
   M = length(segment);
   lmin = fix(g.Options.Pitch.PitchStart*fs);
   lmax = fix(g.Options.Pitch.PitchEnd*fs);

   R = R(M:end);
   RR = R(max(lmin, 1):min(lmax, M));

   MaxValue = max(RR);

   Lag = min(find(RR==MaxValue));
   Lag = Lag+lmin;
   L = Lag;

   %This part checks for peaks at one-half, one-third, one-fourth,
one-fifth,
   % and one-sixth of the first estimate of the pitch period.
   % If L/2 (rounded up) is within th pitch range, the maximum value
of the autocorrelation
   % within (L/2)-5 to (L/2)+5 is located. If (L/2)-5 is less than
lmin, lmin is chosen as the
   % lower limit instead of (L/2)-5. If the new peak is greater than
one-half of the old peak,
   % the new corresponding lag replaces the old corresponding lag L.

   if (lmin <= round(L/2))&(round(L/2)<=lmax)
         seg = R(fix(max(lmin,L/2-5)):fix(L/2+5));
         peak = max(seg);
         %check for pitch period doubling error
         if peak > (1/2)*R(L)
            L = L/2;
            %check for pitch period double doubling error (fourfold
errors)
            if (lmin <= round(L/2))&(round(L/2)<=lmax)
                  seg = R(fix(max(lmin,L/2-5)):fix(L/2+5));
                  peak = max(seg);
                  if peak > (1/2)*R(L)
                     L = L/2;
                     break
                  end
            end
            %check for pitch period errors of sixfold
            if (lmin <= round(L/3))&(round(L/3)<=lmax)
                  seg = R(fix(max(lmin,L/3-5)):fix(L/3+5));
                  peak = max(seg);
                  if peak > (1/2)*R(L)
                     L = L/3;
                     break
                  end
```

```
            end

        end
        %check for pitch period tripling errors
        if (lmin <= round(L/3))&(round(L/3)<=lmax)
              seg = R(fix(max(lmin,L/3-5)):fix(L/3+5));
              peak = max(seg);
              if peak > (1/2)*R(L)
                 L = L/3;
                break
            end
        end
        %check for pitch period errors of fivefold
        if (lmin <= round(L/5))&(round(L/5)<=lmax)
              seg = R(fix(max(lmin,L/5-5)):fix(L/5+5));
              peak = max(seg);
              if peak > (1/2)*R(L)
                 L = L/5;
                 break
             end
        end

    end
    %overlab with g.Options.Pitch.Overlap.
    j = j +
round(g.Options.Pitch.SegmentLength*g.Options.Pitch.Overlap*fs);

    LL = [LL, L]

end

L = sum(LL)/N

%The pitch period calculated through the following formula.
pitch = fs/L;

%==================================================================
axes(g.Figure.Axes2Handle)

plot(R,'color',[0 0 0]);

msgbox(['Pitch frequency is ',num2str(fs/L),' Hz'],'Pitch Frequcney
in Hz','help');

function PitchCurve
global g;
%select a voiced area from the speech waveform.
fs = g.signal.fs;

%divide this area to an equal segments each segment has the length
of 51.2 ms.
N =
floor(length(g.signal.Data)/(g.Options.Pitch.SegmentLength*g.Options
.Pitch.Overlap*fs));

%Filter the  selected segment with a butterworth filter having a
cutoff frequency of 600 Hz
[B,A] =
feval(g.Options.Pitch.FilterType,g.Options.Pitch.NumFiltPoles,g.Opti
ons.Pitch.FilterCuttoffFrequency/(fs/2));
filtered = filter(B,A,g.signal.Data);
```

```
%calculate pitch lag for each segment then take the average
j=1;
LL=[];

window =
feval(g.Options.Pitch.WindowType,round(g.Options.Pitch.SegmentLength
*fs));

for i = 1:N-1
   %Compute the autocorrelation function for the selected segment,
then find the maximum value
   %inside the pitch range from 30 to 200 ms.
   %The index of this value will be the lag (L) after adding lmin.
   segment = filtered(j:round(g.Options.Pitch.SegmentLength*fs)+j-
1).*window;

   R = xcorr(segment);
   M = length(segment);
   lmin = fix(g.Options.Pitch.PitchStart*fs);
   lmax = fix(g.Options.Pitch.PitchEnd*fs);

   R = R(M:end);
   RR = R(max(lmin, 1):min(lmax, M));

   MaxValue = max(RR);

   Lag = min(find(RR==MaxValue));
   Lag = Lag+lmin;
   L = Lag;

   %This part checks for peaks at one-half, one-third, one-fourth,
one-fifth,
   % and one-sixth of the first estimate of the pitch period.
   % If L/2 (rounded up) is within th pitch range, the maximum value
of the autocorrelation
   % within (L/2)-5 to (L/2)+5 is located. If (L/2)-5 is less than
lmin, lmin is chosen as the
   % lower limit instead of (L/2)-5. If the new peak is greater than
one-half of the old peak,
   % the new corresponding lag replaces the old corresponding lag L.

   if (lmin <= round(L/2))&(round(L/2)<=lmax)
         seg = R(fix(max(lmin,L/2-5)):fix(L/2+5));
         peak = max(seg);
         %check for pitch period doubling error
         if peak > (1/2)*R(L)
            L = round(L/2);
            %check for pitch period double doubling error (fourfold
errors)
            if (lmin <= round(L/2))&(round(L/2)<=lmax)
                  seg = R(fix(max(lmin,L/2-5)):fix(L/2+5));
                  peak = max(seg);
                  if peak > (1/2)*R(L)
                     L = L/2;
                  end
            end
            %check for pitch period errors of sixfold
            if (lmin <= round(L/3))&(round(L/3)<=lmax)
                  seg = R(fix(max(lmin,L/3-5)):fix(L/3+5));
                  peak = max(seg);
```

```
                    if peak > (1/2)*R(L)
                       L = L/3;
                    end
                end

            end
            %check for pitch period tripling errors
            if (lmin <= round(L/3))&(round(L/3)<=lmax)
                  seg = R(fix(max(lmin,L/3-5)):fix(L/3+5));
                  peak = max(seg);
                  if peak > (1/2)*R(L)
                     L = L/3;
                  end
            end
            %check for pitch period errors of fivefold
            if (lmin <= round(L/5))&(round(L/5)<=lmax)
                  seg = R(fix(max(lmin,L/5-5)):fix(L/5+5));
                  peak = max(seg);
                  if peak > (1/2)*R(L)
                     L = L/5;
                  end
            end

   end
   %overlab with g.Options.Pitch.Overlap.
   j = j +
round(g.Options.Pitch.SegmentLength*g.Options.Pitch.Overlap*fs);

   LL = [LL, L];

end

L = LL;

%The pitch period calculated through the following formula.
pitch = fs./L;

%======================================================================

axes(g.Figure.Axes2Handle)
x=(1:N-1)*g.Options.Pitch.SegmentLength*g.Options.Pitch.Overlap;


plot(x,pitch,'.','color',[0 0 0]);

xlabel('Time in
second','FontSize',10,'FontUnit','normalized','FontWeight','normal')
;
ylabel('Pitch in
Hz','FontSize',10,'FontUnit','normalized','FontWeight','normal');
title('Pitch
Curve','FontSize',10,'FontUnit','normalized','FontWeight','normal');
```

***Pitch Option Setting***

```
function PitchCuttoffSeting
global g;
Edit5Handle = findobj('Tag','Edit5');
C = get(Edit5Handle,'String');
g.Options.Pitch.FilterCuttoffFrequency = eval(C);

function PitchEndSeting
```

```
global g;
Edit2Handle = findobj('Tag','Edit2');
E = get(Edit2Handle,'String');
g.Options.Pitch.PitchEnd = eval(E);

function PitchFilterTypeSeting
global g;
Popupmenu1Handle = findobj('Tag','PopupMenu1');
val = get(Popupmenu1Handle,'Value');
%close(gcf)
if val == 1
   g.Options.Pitch.FilterType = 'butter'
elseif val == 2
   g.Options.Pitch.FilterType = 'buttord'
elseif val == 3
   g.Options.Pitch.FilterType = 'besself'
elseif val == 4
   g.Options.Pitch.FilterType = 'chepy1'
elseif val == 5
   g.Options.Pitch.FilterType = 'ellip'
end

function PitchNPolesSeting
global g;
Edit4Handle = findobj('Tag','Edit4');
P = get(Edit4Handle,'String');
g.Options.Pitch.NumFiltPoles = eval(P);

function PitchOverlabSeting
global g;
Edit6Handle = findobj('Tag','Edit6');
O = get(Edit6Handle,'String');
g.Options.Pitch.Overlap = eval(O);

function PitchSegmentLengthSeting
global g;
Edit3Handle = findobj('Tag','Edit3');
L = get(Edit3Handle,'String');
g.Options.Pitch.SegmentLength = eval(L);

function PitchStartSeting
global g;
Edit1Handle = findobj('Tag','Edit1');
S = get(Edit1Handle,'String');
g.Options.Pitch.PitchStart = eval(S);


function PitchWindowTypeSeting
global g;
Popupmenu2Handle = findobj('Tag','PopupMenu2');
val = get(Popupmenu2Handle,'Value');

%close(gcf)
if val == 1
   g.Options.Pitch.WindowType = 'hamming'
elseif val == 2
   g.Options.Pitch.WindowType = 'hanning'
elseif val == 3
   g.Options.Pitch.WindowType = 'kaiser'
elseif val == 4
   g.Options.Pitch.WindowType = 'bartlett'
```

```
elseif val == 5
   g.Options.Pitch.WindowType = 'blackman'
end
```

===========================================================

==================*Spectrogram Analysis*====================

```
function spegram
global g;
fs=g.signal.fs;
% preemphasis
%x = filter([1 -0.9],1,g.signal.Data);
%FilteredSpeech
=feval('filter',g.Options.Specgram.FilterCoffecients,1,g.signal.Data
);
FilteredSpeech =filter([1 -
g.Options.Specgram.FilterCoffecients],1,g.signal.Data);
%FilteredSpeech = FilteredSpeech(round(1.92*fs):round(2.25*fs));
spectrogram =
20*log10(abs(SPECGRAM(FilteredSpeech,g.Options.Specgram.FftSize,g.si
gnal.fs,feval(g.Options.Specgram.WindowType,g.Options.Specgram.Windo
wWidth),fix(g.Options.Specgram.WindowWidth*g.Options.Specgram.Overla
p))));
%spectrogram =
20*log10(abs(SPECGRAM(g.signal.Data,g.Options.Specgram.FftSize,g.sig
nal.fs,feval(g.Options.Specgram.WindowType,g.Options.Specgram.Window
Width),g.Options.Specgram.Overlap)));
axes(g.Figure.Axes2Handle)

colormap(hot);
th=-20;
y=spectrogram;
ymax=max(max(y));
ymin=max([th min(min(y))]);
k=find(y<ymin);
y(k)=ymin;

cmlen=length(colormap);

y=(y-ymin)/(ymax-ymin)*(cmlen-1)+1;

%y=flipud(y);
t=(0:length(g.signal.Data)-1)/fs;
f=(0:g.Options.Specgram.FftSize/2-1)/g.Options.Specgram.FftSize*fs;
x=image(t,f,y);
set(gca,'YDir','normal')
xlabel('Time,
Seconds','FontSize',10,'FontUnit','normalized','FontWeight','normal'
);
ylabel('Frequency,
Hz','FontSize',10,'FontUnit','normalized','FontWeight','normal');
title('Spectrogram','FontSize',10,'FontUnit','normalized','FontWeigh
t','normal');
```

### *Spectrogram Options Interface*

```
function fig = SpecgOption()
% This is the machine-generated representation of a Handle Graphics
object
```

```
% and its children.  Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was
saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your
path.
%
% NOTE: certain newer features in MATLAB may not have been saved in
this
% M-file due to limitations of this format, which has been
superseded by
% FIG-files.  Figures which have been annotated using the plot
editor tools
% are incompatible with the M-file/MAT-file format, and should be
saved as
% FIG-files.

load Option

h0 = figure('Color',[0.8 0.8 0.8], ...
    'Colormap',mat0, ...
    'FileName','C:\MATLABR11\project\Option.m', ...
    'Name','Specgram Option', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[403 287 192 186], ...
    'Tag','Fig1', ...
    'ToolBar','none');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[1 1 1], ...
    'Callback','WindowType', ...
    'ListboxTop',0, ...
    'Position',[0.34375 0.2634408602150538 0.4010416666666666
 0.1290322580645161], ...
    'String',['hamming ';'hanning ';'kaiser
 ';'bartlett';'blackman'], ...
    'Style','popupmenu', ...
    'Tag','PopupMenu1', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.7 0.7 0.7], ...
    'ListboxTop',0, ...
    'Position',mat1, ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[0.5104166666666666 0.5860215053763441
 0.4166666666666666 0.1290322580645161], ...
    'String','Window Length', ...
```

```
   'Style','text', ...
   'Tag','Text1');
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[1 1 1], ...
   'ListboxTop',0, ...
   'Position',mat2, ...
   'String','Overlab', ...
   'Style','text', ...
   'Tag','Text4');
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
   'Callback','close(gcf)', ...
   'ListboxTop',0, ...
   'Position',mat3, ...
   'String','OK', ...
   'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[1 1 1], ...
   'Callback','WindowWidthSeting', ...
   'ListboxTop',0, ...
   'Position',mat4, ...
   'Style','edit', ...
   'Tag','Edit1');
h1 = uicontrol('Parent',h0, ...
   'Units','normalized', ...
   'BackgroundColor',[1 1 1], ...
   'Callback','OverlabSeting', ...
   'ListboxTop',0, ...
   'Position',mat5, ...
   'Style','edit', ...
   'Tag','Edit2');
if nargout > 0, fig = h0; end
```

===========================================================

*Spectrogram Options*

```
function SpecgramFftLengthSeting
global g;
Edit2Handle = findobj('Tag','Edit3');
F=get(Edit2Handle,'String');
g.Options.Specgram.FftSize = eval(F);

function SpecgramOverlabSeting
global g;
Edit2Handle = findobj('Tag','Edit2');
O = get(Edit2Handle,'String');
g.Options.Specgram.Overlap = eval(O);



function SpecturmPremphasesSeting
global g;
Edit4Handle = findobj('Tag','Edit4');
P = get(Edit4Handle,'String');
g.Options.Specgram.Premphases = eval(P);
```

```
function SpecgramWindowType
global g;
h=findobj('Tag','PopupMenu1');
val = get(h,'Value');
close(gcf)
if val == 1
   g.Options.Specgram.WindowType = 'hamming'
elseif val == 2
   g.Options.Specgram.WindowType = 'hanning'
elseif val == 3
   g.Options.Specgram.WindowType = 'kaiser'
elseif val == 4
   g.Options.Specgram.WindowType = 'bartlett'
elseif val == 5
   g.Options.Specgram.WindowType = 'blackman'
end

function SpecgramWindowWidthSeting
global g;
Edit2Handle = findobj('Tag','Edit1');
W=get(Edit2Handle,'String');
g.Options.Specgram.WindowWidth = eval(W);
```

---

## *Spectrum Analysis*

```
function Spectrum

global g
fs=g.signal.fs;
%select a segment of data with length of SegmentLength in s.
Term=ginput(1);
Term=fix(Term*fs)
begining = max(Term(1),1);
data=g.signal.Data(begining:begining+round(g.Options.Spectrum.Segmen
tLength*fs)-1);
%data windowing
window=feval(g.Options.Spectrum.WindowType,round(g.Options.Spectrum.
SegmentLength*fs));
data=data.*window;
%compute fft for the segment (spectral representation)
spectrum = abs(fft(data,g.Options.Spectrum.FftLength));
%spectrum smoothing, by computing lpc coffecients and thier spectrum
[a d]=lpc(data,g.Options.Spectrum.CofficientsNumber);
LpcSpectrum=d./abs(fft(a,g.Options.Spectrum.FftLength ));
%make x axis in frequency unit
frequency=(1:fix(g.Options.Spectrum.FftLength
/2))/(0.5*g.Options.Spectrum.FftLength )*fs;
%plot spectrum and LpcSpecturm.

axes(g.Figure.Axes2Handle);
semilogy(frequency,LpcSpectrum(1:end/2),'-.','Color','k')
hold on

semilogy(frequency,spectrum(1:end/2),'Color','k')
hold off
xlabel('Frequency,
Hz','FontSize',10,'FontUnit','normalized','FontWeight','normal');
ylabel('Amplitude, log
scal','FontSize',10,'FontUnit','normalized','FontWeight','normal');
```

```
title('Spectrum
smooting','FontSize',10,'FontUnit','normalized','FontWeight','normal
');
```

==================================================================

## *Spectrum Options Setting*

```
function SpecturmFftLengthSeting
global g;
Edit2Handle = findobj('Tag','Edit2');
F = get(Edit2Handle,'String');
g.Options.Spectrum.FftLength = eval(F);

function SpectrumLpcCoffNumSeting
global g;
Edit1Handle = findobj('Tag','Edit1');
C = get(Edit1Handle,'String');
g.Options.Spectrum.CofficientsNumber = eval(C);

function SpectrumSegmentLengthSeting
global g;
Edit3Handle = findobj('Tag','Edit3');
S = get(Edit3Handle,'String');
g.Options.Spectrum.SegmentLength = eval(S);

function SpectrumWindowTypeSeting
global g;
Popupmenu1Handle = findobj('Tag','PopupMenu1');
val = get(Popupmenu1Handle,'Value');
%close(gcf)
if val == 1
   g.Options.Spectrum.WindowType = 'hamming'
elseif val == 2
   g.Options.Spectrum.WindowType = 'hanning'
elseif val == 3
   g.Options.Spectrum.WindowType = 'kaiser'
elseif val == 4
   g.Options.Spectrum.WindowType = 'bartlett'
elseif val == 5
   g.Options.Spectrum.WindowType = 'blackman'
end
```